May/June 1982 Volume 2, Number 3 $2.95 (USA) £1.50 (UK)

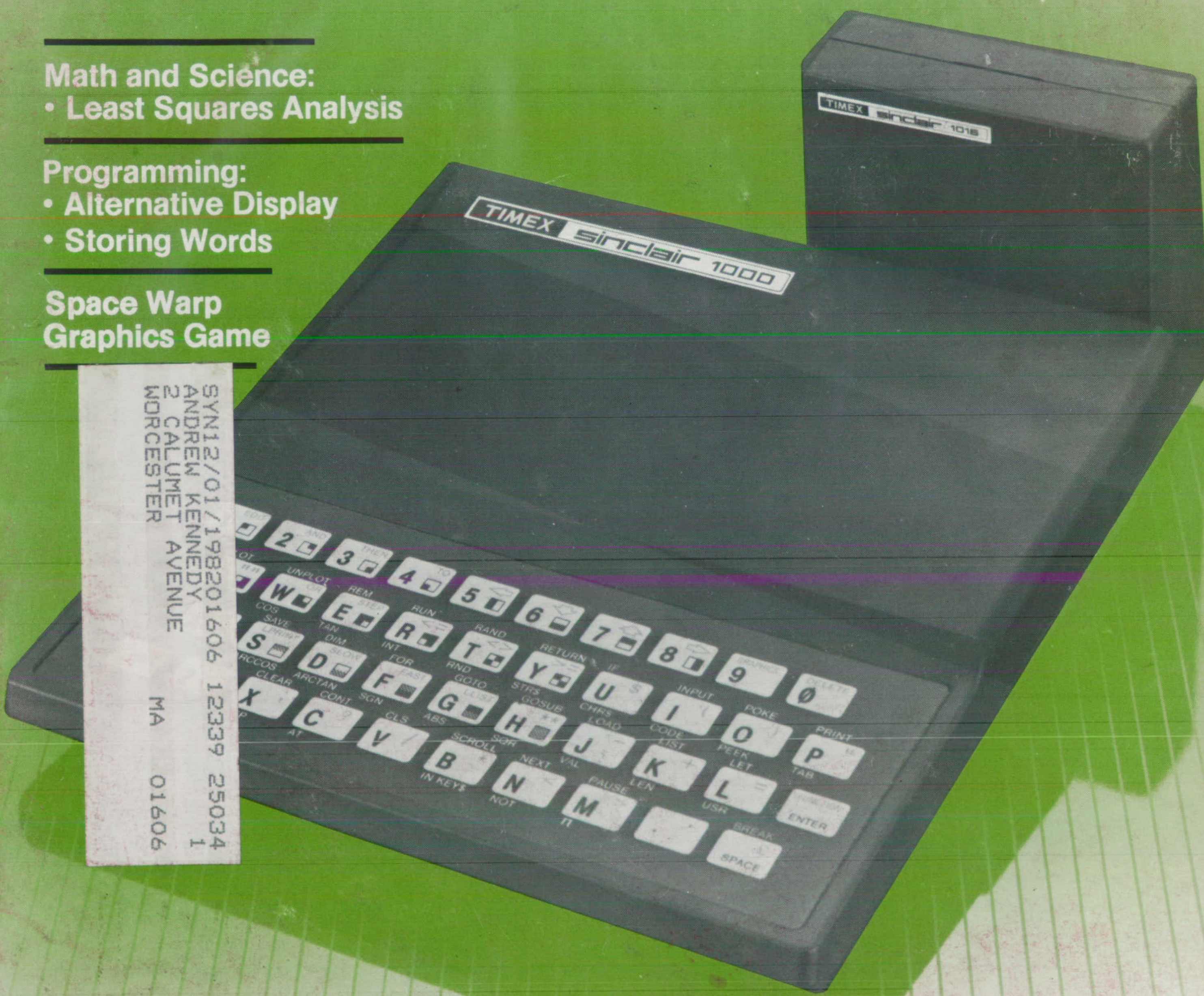
The magazine for Sinclair users

SYNC®

Math and Science:
- Least Squares Analysis

Programming:
- Alternative Display
- Storing Words

Space Warp Graphics Game



SYN12/01/1982 01606 12339 25034 1
ANDREW KENNEDY
2 CALUMET AVENUE
WORCESTER MA 01606

Make your "LITTLE" ZX81 *work like a* BIG *computer with* BYTE-BACK *modules*

INSTANT INFORMATION WITH BYTE-BACK'S MD-1 MODEM *only* $99.95 KIT

WIRED and TESTED $119.95

Use your phone to connect your "LITTLE" ZX81 to the "LARGEST" computer networks in the world. With **BYTE-BACK's MD-1 MODEM** connected all you do is dial a phone number (usually local), press a few keys and watch the data appear on your TV screen. (Software is included)

This **MODEM** can be used in either the "originate" or "answer" mode with selectable baud rate.

You can have immediate access to:

DOW JONES UPI And Much More!

RS-232 PORT INCLUDED

As an extra bonus an RS-232 port is provided to allow you to drive all standard RS-232 peripherials.(75 to 9600 Baud)

16-K MEMORY MODULE (M-16)

It has a Metal Case

only

$69.95 KIT

WIRED and TESTED $79.95

in stock!

But unlike other 16-K memory modules, up to three BYTE-BACK M-16 memory modules can be connected at one time to get a total of 48K. (It can be used with the SINCLAIR memory module.)

64-K MEMORY MODULE

It has a Metal Case

NEW

in stock!

$139.95 KIT

WIRED and TESTED $154.95

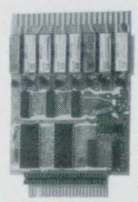
The new **M-64,** together with the **ZX-81,** gives a full 64K of system memory (ROM + RAM) with the same high reliability of our proven 16-K module.

BYTE-BACK'S BB-1 CONTROL MODULE

$59.00 KIT *In Stock!*

WIRED and TESTED $69.

- **8 Independent Relays** *(with LED status indicators)*
- **8 Independent TTL Inputs** *(with Schmitt trigger buffers)*



- By using a single POKE command you can change and latch the status of each of the 8 relays.
- Your **ZX80/1** can read the status of all 8 inputs by the use of a single PEEK command.
- A comprehensive manual is included that has complete application details.

ALL MODULES CARRY 90-DAY WARRANTY

Remember with: **BYTE BACK** *modules you are NOT limited to using only one module at a time!*

BYTE-BACK CO.
RT. 3, BOX 147 • BRODIE RD.
LEESVILLE, S.C. 29070
Ph. (803) 532-5812

☐ M-64 Kit	$139.95	☐ M-16 Kit	$69.95
☐ M-64 Wired and Tested	$154.95	☐ M-16 Wired and Tested	$79.95
☐ M-64 Blank PC Board	$19.95	☐ M-16 Blank PC Board	$19.95
☐ BB-1 Kit and Manual	$59	☐ Modem Kit	$99.95
☐ BB-1 Wired and Tested and Manual	$69	☐ Modem Wired and Tested	$119.95
☐ BB-1 Blank PC Board and Manual	$29		

Shipping and Handling $4.95

ORDER PHONE (803) 532-5812 **Bill My** ☐ Visa ☐ Mastercharge

Exp. Date ____________ Card No. ____________

Name ____________

Address ____________

City/State/Zip ____________

Mail To: **BYTE-BACK CO. • Rt. 3, Box 147 • *Brodie Rd.* • Leesville, S.C. 29070**

The magazine for Sinclair users

SYNC®

May/June 1982 Volume 2, Number 3

DEPARTMENTS

HARDWARE AND THE KEYBOARD

PROGRAMMING TECHNIQUES AND FUNCTIONS

MATH AND SCIENCE

GAMES

REVIEWS

Staff

Volume 2, Number 3

SYNC (USPS: 585-490: ISSN: 0279-5701) is published bi-monthly for $16 per year by Creative Computing, 39 E. Hanover Ave., Morris Plains, NJ 07950.

Second class postage paid at New York, NY 10001, and at additional mailing offices.

Subscription rates: USA: 6 issues, $16; 12 issues, $30: 18 issues, $42. Canada: $3 per year additional. Other foreign: $5 per year additional. U.K. air: 6 issues, £13. Minimum charge card order: $10.

For *SYNC* advertising information, contact Jim Beloff, *SYNC* Advertising Sales Manager, Ziff-Davis Publishing Company, One Park Ave., New York, NY 10016 (phone 212/725-4216).

All other correspondence should be addressed to: *SYNC*, 39 E. Hanover Ave., Morris Plains, NJ 07950. In U.K., *SYNC*, 27 Andrew Close, Stoke Golding, Nuneaton CV136EL.

Postmaster: Send address changes to *SYNC*, PO Box 789-M, Morristown, NJ 07960.

MEMBER MITA

A creative computing PUBLICATION

letters

Help Wanted

Dear Editor:

I have a special request. I would like to have some information on producers and shops which sell ZX81 music programs, like one which changes the ZX81 into an electric organ or a synthesizer. I own a ZX81 and I am very interested in making and playing music on it. Also I am interested in a good accounting system for a small business.

Arnar Matthiasson
Samvinnuskolinn
Bifrost
311 Borgarnes
Iceland

Dear Editor:

I recently acquired my ZX81 (16K) with little knowledge of programming under my belt. I received my first issue of *SYNC* (2:1), and I was excited at the prospect of using machine language. Then I encountered two problems:

1) I would be interested in knowing how to modify the "Flattop Lander" code for my machine. The system variables appear to be different, and I have no source equivalents.

2) I twice, no thrice, entered the "Life" program with no success. The initial border routine worked, but after I entered some numbers/cells the system would crash when I input 0 to continue. Are there any bugs? I even entered it once using Mr. Ornstein's *Window* program (a long process I can assure you).

Any help you could give me would be appreciated. I enjoy your magazine, though I wish that it were monthly. Also, which back issues are still available?

Keith Liggeti
21111 Strathmoor Lane
Huntington Beach, CA 92646

Ed.—The "Life" program did have some problems as some readers have called to our attention. The author has supplied the revisions which are included in our Glitchoidz Report. A few back issues of **SYNC** *1:2 are available at $2.50 each. Payment should accompany the order.*

Dear Editor:

Enclosed is a list of several errors that I found in *The Gateway Guide to the ZX81 and ZX80*. I am sure that most readers would spot the errors quickly, but beginners may not. I feel that the book is well worth the money, as is *The ZX81 Companion* (but I wish more about the intricate working of the ROM had been included).

Now a question. Is there any way to write a machine language routine to read TRS-80 500 BAUD tapes? I sent for a kit to do so, but it is really designed for an S-100 Bus operating with a Z80 CPU. How about an article in *SYNC* on this? How about more articles on machine routines to extend the 8K Basic like the READ routine in Edward Kennedy's article (*SYNC* 1:5). How about MERGE? How about a routine to set a single pixel?

I liked very much David Ornstein's article on the 4K ROM SAVE command. I hope that he does as threatened and writes an article on a full fledged cassette based I/O processing system, especially if he includes details on wiring the cassette to forward or reverse under CPU control.

I have also sent for a light pen (designed for the TRS-80) that I would like to hook up to my 8K ROM ZX80. Any suggestions? How about some articles on adding output ports to make the ZX80 compatible with TRS-80 peripherals?

Timothy McIlwee
Flagler Palm Coast High
PO Box 488
Bunnell, FL 32010

Ed.—The corrections have been passed on to our Book Division. Again our readers have set out a number of challenges. We look forward to your responses.

Program Problems

Dear Editor:

I am the owner of a ZX80 and a ZX81. I have been trying some of the examples in your magazine and have been having problems. E.g., In *Robot Composer* (1:5) I cannot enter line 150; in *Defuse* (1:5), lines 10, 20, 30, 40; in *Graphic Surprises* (1:3), lines 130, 200, 1010; *Variable Conversions* (1:3), line 30; The *TL$ Function (1:4)*, line 50 LET P$=TL$(A$) (syntax error); *Mini-Billboard*, line 21 LET A$=TL$(A$). Please tell me how I can do these.

Edward A. Parker
582 East Sunset Dr.
Altadena, CA 91001

Ed.—First and foremost be sure that you are entering your program on the correct ROM. All the programs in issues 1:1, 1:2, 1:3, and 1:4 are for the 4K ROM and cannot be entered on the 8K ROM without translation. In issues from 1:5 on programs for both ROMs are included, and the machine requirements for the programs are given at the top of the page. Most of the lines referred to above seem to involve the random feature. On the 4K ROM RANDOMISE is on the J key and RND(x) must be spelled out; on the 8K ROM INT and RND are functions and must be entered directly from the keyboard in one keystroke. In the line P$=TL$(P$), ignore the syntax error and finish typing in the line. It serves to remind you that something must be completed, e.g., the second parenthesis or quotation marks. For the 8K ROM the line must be rewritten as LET P$=P$(2 TO #).

ZX80 Slow Mode

Dear Editor:

Is there any way that I can get my ZX80 with the 8K ROM to run in the SLOW mode?

Anthony Larry
228 Main St.
West Haven, CT 06516

Ed.—At present the only method we have heard of is the MicroAce Video Upgrade Kit (see a review of this kit in **SYNC** *2:1, p. 27). However, MicroAce has closed down its U.S. operation, and we have received no information whether a U.S. distributor for the kit has been arranged. MicroAce can be reached at: MicroAce Compshop, 14 Station Road, New Barnet, Hertfordshire EN5 1QW, United Kingdom.*

Software Publishers—Analyze the NEED!

The TIMEX and SINCLAIR ZX systems are cassette-based. Everything depends on delivering software product the user can LOAD very readily, without difficulty. In publishing software for these machines, reputation and repeat orders depend on good programming AND ready-loading product.

Software Publishers—Qualify the SOURCE!

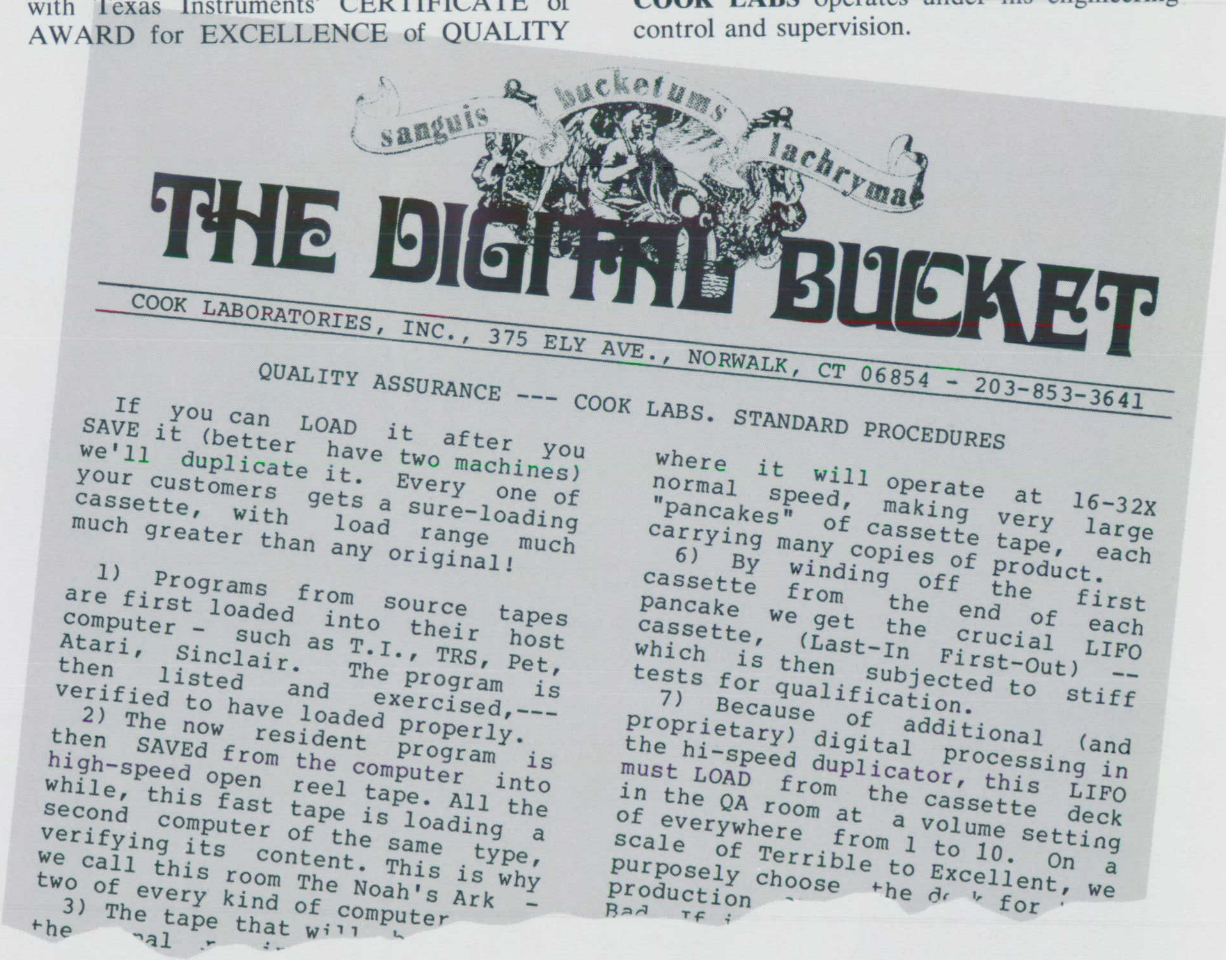
COOK LABS is the product source that DOES duplicate digitally — not audio-analog.

COOK LABS has been making digital cassettes for serious software publishers for many years. Why?

COOK LABS is the only duplicating source with Texas Instruments' CERTIFICATE of AWARD for EXCELLENCE of QUALITY (1981). Users don't have problems with T.I. cassette software. They needn't with yours.

SINCE 1952 Emory Cook's name has been internationally associated with leadership in electronics design.

COOK LABS operates under his engineering control and supervision.



sanguis bucketums lachryma

THE DIGITAL BUCKET

COOK LABORATORIES, INC., 375 ELY AVE., NORWALK, CT 06854 - 203-853-3641

QUALITY ASSURANCE --- COOK LABS. STANDARD PROCEDURES

If you can LOAD it after you SAVE it (better have two machines) we'll duplicate it. Every one of your customers gets a sure-loading cassette, with load range much much greater than any original!

1) Programs from source tapes are first loaded into their host computer - such as T.I., TRS, Pet, Atari, Sinclair. The program is then listed and exercised,--- verified to have loaded properly.

2) The now resident program is then SAVEd from the computer into high-speed open reel tape. All the while, this fast tape is loading a second computer of the same type, verifying its content. This is why we call this room The Noah's Ark - two of every kind of computer

3) The tape that will b...

...where it will operate at 16-32X normal speed, making very large "pancakes" of cassette tape, each carrying many copies of product.

6) By winding off the first cassette from the end of each pancake we get the crucial LIFO cassette, (Last-In First-Out) -- which is then subjected to stiff tests for qualification.

7) Because of additional (and proprietary) digital processing in the hi-speed duplicator, this LIFO must LOAD from the cassette deck in the QA room at a volume setting of everywhere from 1 to 10. On a scale of Terrible to Excellent, we purposely choose the d... for ... production ...

SOFTWARE PUBLISHERS — PHONE, WRITE, USE COUPON . . . GET INFORMATION NOW!

COOK LABORATORIES, INC.
P.O. Box 529
Norwalk, CT 06856 (Phone 203-853-3641)

☐ Inform me about COOK services for:
- ☐ Software duplication.
- ☐ Procurement of labels, inserts, etc.
- ☐ Fulfillment, including fulfillment of individual orders.

☐ Send free booklet: "Cassettes — An Effective Software Medium."

NAME ______________________

COMPANY ______________________

ADDRESS ______________________

______________________ ZIP ________

PHONE (OPTIONAL) ______________________

The Hidden Chessmen

Dear Editor:

I have received several requests to convert *The Hidden Chessmen* (*SYNC* 1:6) to use on a ZX81. Presently, I have an 8K ROM in my ZX80 and a 16K RAM pack. The following changes were needed to get the program running:

```
 1 LET B=1+INT(8*RND)
 2 LET C=1+INT(8*RND)
 3 LET K=1+INT(8*RND)
 4 LET L=1+INT(8*RND)
 6 LET R=1+INT(8*RND)
 7 LET S=1+INT(8*RND)
44 LET X=INT(Q/10)
50 LET Q=PEEK 16396+256*
PEEK 16397+2*X-1+66*(Y-1)
62 IF . . . THEN POKE(Q+33),48
64 IF . . . THEN POKE(Q+34),55
```

The program takes more than 1K RAM, but I think it will fit in 2K.

Roger Haar
19372 Holts Rd.
Martin, OH 43445

Ed.—Our thanks to Roger Haar for these changes to make the program available to our 8K ROM readers.

sync notes

Paul Grosjean

Timex Sinclair 1000

The Timex Computer Corporation will enter the personal computer market with the Timex Sinclair 1000. Sales will begin in July through the more than 100,000 Timex retail outlets, including computer stores, department stores and chains, consumer electronics, jewelry, and drug stores. This step is the result of an agreement between Timex and Sinclair Research Ltd. for Timex to market computers using Sinclair's technical expertise. Sinclair's mail order sales of the ZX81 will be phased out as Timex begins its marketing program. The announcement of the Timex entry into this new field and of the agreement with Sinclair was made by Daniel D. Ross, Vice President of Timex Computer Corporation, an affiliate of Timex Corporation, in New York on April 20.

The Timex Sinclair 1000 is basically the Sinclair ZX81, which Timex has already been manufacturing in Dundee, Scotland, but with two major differences. First, the new machine will have 2K RAM instead of the current 1K on the ZX81. Second, the new machine will sell for a suggested retail price of $99.95 instead of the $149.95 for the ZX81. Also featured will be an instruction manual especially written for the first time computer user with step by step instruction and a course in fundamental programming.

Peripherals for the Timex Sinclair 1000 will also be sold by Timex along with the basic machine. The first one available is the 16K RAM expansion module for $49.95 (Sinclair's current 16K RAM is $99.95). In the fourth quarter of 1982 a printer and a telephone modem are expected to go on sale for $99.95 each. The modem will offer a significant enhancement for many users and will feature: 300 BAUD, standard Bell Telephone jack attachments, and auto-dial capability. It will give a direct tie-in with large computer data services.

Timex plans to supplement the hardware offerings with a range of software, including business, personal financial management, education, and entertainment. The price range of the programs is expected to be from $9.95 to $19.95.

The marketing program will aim at the first-time computer buyer, the educational market, and computer buffs. The potential market for personal computers is estimated at over 90,000,000 customers. In addition to instructional displays in the retail outlets, Timex plans to support the retailers with an extensive service network, a 90-day guarantee, and a national ad campaign beginning in August.

SYNC Coverage

SYNC will expand its coverage of the Sinclair type computers by adding the Timex Sinclair 1000 to the list. We will continue to help you get more out of your computer by providing instruction, entertainment and product information.

Do You Want to...

Advertise in SYNC?

Contact Jim Beloff, SYNC Advertising Sales Manager, Ziff-Davis Publishing Co., One Park Ave., New York, NY 10016; phone: (212) 725-4216. Jim now handles all aspects of advertising in SYNC. He will be glad not only to supply you with all information about ad rates, specifications time schedules, and copy submission but also to answer any questions you may have about "getting in SYNC."

Be listed in the Resources Column?

Send a brief announcement of your new product or service or users group similar to the ones you have already seen in the Resources Column to "SYNC Resources." There is no charge for a one time listing. Repeat listings may be made at our classified rate which goes by the inch. Be sure to include the name of your product, brief details about it, and exact details on how readers may order.

Send to: SYNC Magazine, 39 E. Hanover Ave., Morris Plains, NJ 07950.

Write for SYNC?

In our Jan/Feb 1982 issue we ran an article "Writing for SYNC." If you want to submit an article to SYNC and did not get this issue, send a self-addressed stamped envelope to the managing editor for a copy of the article.

If you are planning a rather lengthy article, it would be best to check whether we would be interested in the topic before you put the work into writing it up. Just drop a note with a brief outline or summary of what you have in mind to the managing editor. It is not unusual for us to receive several articles or programs on similar topics within a matter of a few weeks of each other.

Again we want to emphasize that manuscripts must be typed and double spaced. If you are submitting a printout of your article, *please set your printer on double space*. Program listings, however, should be *single spaced* and should show the lines *just as they are on the computer screen*. A camera ready program listing adds significantly to your article. When printers are available, printouts will be preferred. Long programs should also be submitted on tape.

If you are submitting a program which is not a direct printout from your ZX80/81 computer, please enter your program from your listing before you send it to us.

Send to: SYNC Magazine, 39 E. Hanover Ave., Morris Plains, NJ 07950.

Order products from our advertisers?

1) Allow sufficient time for an order to be processed, shipped, and delivered. Understandably ZX80/81 users are eager to get their orders, but patience must be urged. Two to three weeks is simply not enough turnaround time in most cases, especially if the supplier waits for a check to clear.

2) If you want to order internationally, you can pay in the following ways: a) a personal check (remember that a bank only accepts a check for collection; when you get your money immediately the bank is really giving you the money from your own account and will deduct that amount if the check bounces; the check must clear before it is actually credited; so your supplier may wait to process your order until the check clears); b) a bank draft (such as when you make a withdrawal from a savings account); c) a "foreign draft" in the money of the country you are ordering from (the charge in our area is about $10); d) an international money order (in dollars) purchased at your bank

usually for a flat fee (in our area about $1.25 regardless of the amount); e) an international money order purchased at the Post Office (the rates vary according to the amount of the money order and the country to which it goes; for England the cost is $1.30 for up to $25 and $1.50 for $25-50).

3) Remember that banks charge the depositor for handling a foreign check; in our area the charge can be up to $3; the same holds for foreign banks handling U.S. checks. Needless to say, such a charge can eat up a seller's profit on a small order. Some will include it in either a handling charge or higher prices)

4) A check with local customs reveals that in general there is no duty on software but that there might be on hardware. To be on the safe side, check with your local customs office before ordering, especially if the cost or quantity of items is substantial. If there is a customs assessment, the Post Office will notify you that you have a package with the amount due which includes a charge of about $2 by the Post Office for handling the transaction.

Ask technical questions about your ZX80/81?

Write a letter to the editor stating your problem or question. We will refer the letter to some of our authors who have agreed to answer questions. (Please do not call SYNC because the people who can answer the questions are not at our offices.) We will print the question and answer whenever possible. The chances are good that you are not the only one who has the question.

Form a users group or publicize one?

Send details to our Resources Column.

Developing a Memory Map Consensus

Nick Lambert of Quicksilva has proposed that suppliers for the Sinclair computers reach a consensus on some kind of memory map to avoid "a whole heap of problems" to everyone's benefit. He has proposed the following ZX Computer Memory Map on a provisional basis for discussion. The comments of suppliers are welcome and should be sent to Nick Lambert, Quicksilva, 95 Upper Brown Hill Road, Maybush, Southampton, Hampshire, U.K.

SYNC Program Listings

Readers should note the following conventions used in the program listings in this issue:

— The number symbol is used in PRINT statements to show necessary spaces.

"A"— The underline is used in PRINT statements to indicate graphics. Press the key designated in whatever way your machine requires to get the graphic.

INPUT — In PRINT statements when a word or token is underlined enter the word from the keyboard rather than type it in letter by letter. This is a memory saving technique.

After typing in a program line, you must put the line into your program. On the ZX80 and ZX81 with 8K ROM this is done by hitting the NEWLINE key. On the ZX81 the same thing is done by hitting ENTER. The words are used interchangeably.

A number followed by a b is binary; a number followed by h is hex.

ZX Computer Memory Map

64K–48K	Screen Display Area	**User RAM (Screen routine only)** The user RAM appears in this block where the screen display routines and hardware use it to output the display. D-FILE can appear anywhere in this block so it really cannot be used for any other purpose.
48K–32K	More I/O and Memory Blocks Software Switches I/O Space QUICKSILVA Hi-res memory	**Read/write only memory** Due to the internal hardware of the computer no opcode fetches can be made from this block. So you cannot run any machine code from here. You can read and write to memory though so it is fine for data storage. Also it seems to be the best place to do Input/Output from. Unless there is an extremely good reason not to, Quicksilva intend to put the 6K of memory needed for our Hi-res board starting at 32K. This will enable us to make the Hi-res software run significantly faster. Other areas within this block are open to debate.
32K–16K	User RAM	**User RAM** Most people are, or will be, using a 16K RAM pack of some kind so really this block should be left completely for this purpose. No ROM or I/O from this block.
16K–8K	? 14K Voice 12K Graphics and Extended Basic 10K D.O.S. and Network	**ROMs (for add-on boards)** This 8K block is the only area left, apart from user RAM space from which you can do an opcode fetch. For this reason this block should be saved for extra ROMs. Actual functions of the 2K areas in this block can be swapped around of course. However, if two companies are doing the same product which requires some of this ROM space, to avoid wasting valuable space, it would be sensible for both to use the same 2K area. So here it would be advisable to settle for a definite layout, i.e., 8K to 10K for Disc Operating Systems, etc.
8K–0K	Basic	**ROM (Sinclair Basic ROM)**

The $149[95] personal computer.

Introducing the Sinclair ZX81

If you're ever going to buy a personal computer, now is the time to do it.

The new Sinclair ZX81 is the most powerful, yet easy-to-use computer ever offered for anywhere near the price: only $149.95* completely assembled.

Don't let the price fool you. The ZX81 has just about everything you could ask for in a personal computer.

A breakthrough in personal computers

The ZX81 is a major advance over the original Sinclair ZX80—the world's largest selling personal computer and the first for under $200.

In fact, the ZX81's new 8K Extended BASIC offers features found only on computers costing two or three times as much.

Just look at what you get:

■ Continuous display, including moving graphics

■ Multi-dimensional string and numerical arrays

■ Mathematical and scientific functions accurate to 8 decimal places

■ Unique one-touch entry of key words like PRINT, RUN and LIST

■ Automatic syntax error detection and easy editing

■ Randomize function useful for both games and serious applications

■ Built-in interface for ZX Printer

■ 1K of memory expandable to 16K

The ZX81 is also very convenient to use. It hooks up to any television set to produce a clear 32-column by 24-line display. And you can use a regular cassette recorder to store and recall programs by name.

*Plus shipping and handling. Price includes connectors for TV and cassette, AC adaptor, and FREE manual.

If you already own a ZX80

The 8K Extended BASIC chip used in the ZX81 is available as a plug-in replacement for your ZX80 for only $39.95, plus shipping and handling—complete with new keyboard overlay and the ZX81 manual.

So in just a few minutes, with no special skills or tools required, you can upgrade your ZX80 to have all the powerful features of the ZX81. (You'll have everything except continuous display, but you can still use the PAUSE and SCROLL commands to get moving graphics.)

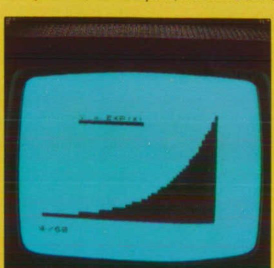
With the 8K BASIC chip, your ZX80 will also be equipped to use the ZX Printer and Sinclair software.

Order at no risk**

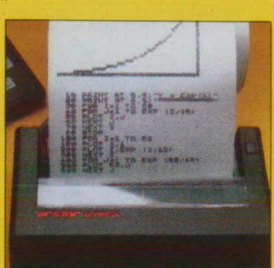
We'll give you 10 days to try out the ZX81. If you're not completely satisfied, just return it to Sinclair Research and we'll give you a full refund.

And if you have a problem with your ZX81, send it to Sinclair Research within 90 days and we'll repair or replace it at no charge.

**Does not apply to ZX81 kits.

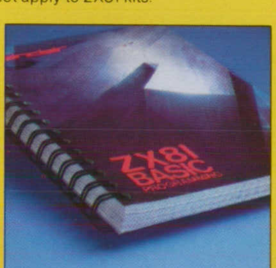


NEW SOFTWARE: Sinclair has published pre-recorded programs on cassettes for your ZX81, or ZX80 with 8K BASIC. We're constantly coming out with new programs, so we'll send you our latest software catalog with your computer.



ZX PRINTER: The Sinclair ZX Printer will work with your ZX81, or ZX80 with 8K BASIC. It will be available in the near future and will cost less than $100.

16K MEMORY MODULE: Like any powerful, full fledged computer, the ZX81 is expandable. Sinclair's 16K memory module plugs right onto the back of your ZX81 (or ZX80, with or without 8K BASIC). Cost is $99.95, plus shipping and handling.



ZX81 MANUAL: The ZX81 comes with a comprehensive 164-page programming guide and operating manual designed for both beginners and experienced computer users. A $10.95 value, it's yours free with the ZX81.

The $99^95 personal computer.

Introducing the ZX81 kit

If you really want to save money, and you enjoy building electronic kits, you can order the ZX81 in kit form for the incredible price of just $99.95.* It's the same, full-featured computer, only you put it together yourself. We'll send complete, easy-to-follow instructions on how you can assemble your ZX81 in just a few hours. All you have to supply is the soldering iron.

How to order

Sinclair Research is the world's largest manufacturer of personal computers.

The ZX81 represents the latest technology in microelectronics, and it picks up right where the ZX80 left off. Thousands are selling every week.

We urge you to place your order for the new ZX81 today. The sooner you order, the sooner you can start enjoying your own computer.

To order, simply call our toll free number, and use your MasterCard or VISA.

To order by mail, please use the coupon. And send your check or money order. We regret that we cannot accept purchase orders or C.O.D.'s.

CALL 800-543-3000. Ask for operator #509. In Ohio call 800-582-1364. In Canada call 513-729-4300. Ask for operator #509. Phones open 24 hours a day, 7 days a week. Have your MasterCard or VISA ready.

These numbers are for orders only. For information, you must write to Sinclair Research Ltd., 2 Sinclair Plaza, Nashua, NH 03061.

sinclair

AD CODE 05SY	PRICE†	QTY.	AMOUNT
ZX81	$149.95		
ZX81 Kit	99.95		
8K BASIC chip (for ZX80)	39.95		
16K Memory Module (for ZX81 or ZX80)	99.95		
Shipping and Handling	4.95		$4.95
		TOTAL	

MAIL TO: Sinclair Research Ltd., One Sinclair Plaza, Nashua, NH 03061.

NAME ______________________

ADDRESS ______________________

CITY/STATE/ZIP ______________________

† U.S. Dollars

just for fun

In general *SYNC* prefers articles in some depth so that we can help you develop your programming skills and get more out of your computer. However, a number of readers have shared with us some of their favorite short programs which are too long for our "Try This" column and too short for the kind of tutorial that we often use. Some of these programs illustrate a point or demonstrate a technique that the reader has found helpful. Others do something the reader has found interesting. So we have collected a number of these short programs "Just for Fun." If you learn something, great. If you have some fun, great. If you have some that you want to share, send them in. We will have a "Just for Fun" column as often as we have the material.

Renumbering by a USR Routine

Jon Passler

Enter the following program:

```
1 REM    15#5$RNDYVE7YTAB
"RND7Y COS P7P77777TAB" RND
2 REM
3 LET L=USR 16514
4 STOP
```

See "SYNC Notes" for the conventions used here. In addition, the overline indicates inverse characters.

Then in the immediate mode (i.e., without a line number) enter

```
POKE 16531,112
POKE 16533,113
```

RUN the program and note the changes in the listing. Then add five more of the graphics on the 7 key after the five already given in 1 REM and run the program again. Then in the immediate mode:

```
POKE 16515,10
```

and RUN again.

Delete all the lines except 1 REM and save it for future use. It will SAVE and LOAD more quickly without the 16K RAM pack attached. The subroutine can be called in the immediate mode by

```
LET L=USR 16514
```

Note:

```
     LD BC,5
     LD HL,16525
     LD A,59
     ADD A,A
LOOP INC HL
     CP (HL)
     JPNZ, LOOP
     INC HL
     CP (HL)
     RET Z
     LD (HL),B
     INC HL
     LD (HL),C
     INC BC
     INC BC
     INC BC
     INC BC
     INC BC
     JPNZ, LOOP
```

Jon Passler, 344 Cabot St., Beverly, MA 01915.

Sweeper

Bernard Bush

Some uses for UNPLOT, TAB, and SCROLL that are not given directly in the manual can be extremely useful:

1) UNPLOT can be used in a FOR NEXT LOOP to keep the print position on a given line.

2) There are two PLOT (or UNPLOT) positions for a given line. UNPLOT 0,42 or UNPLOT 0,43 will both work for the top line.

3) SCROLL can also be followed by UNPLOT to keep the print position on a given line. Without it SCROLL moves the print position to the bottom left side of the screen.

4) SCROLL can be used in a loop to SCROLL several lines. E.g.,

```
10 FOR J=1 TO 10
20 SCROLL
30 NEXT J
```

The following programs show some uses for UNPLOT and TAB.

```
SWEEPER

 10 LET A=INT (RND*3)+1
 20 LET B=INT (RND*20)+5
 30 IF A=1 THEN LET A$="■>NO"
 40 IF A=2 THEN LET A$=" HI"
 50 IF A=3 THEN LET A$=" O"
 60 FOR K=1 TO B
 70 PRINT TAB K;A$;
 80 GOSUB 190
110 NEXT K
120 FOR J=30 TO 1 STEP -1
130 PRINT TAB J;"/GO "
140 GOSUB 190
170 NEXT J
180 GOTO 10
190 IF A=1 THEN UNPLOT 0,42
200 IF A=2 THEN UNPLOT 0,40
210 IF A=3 THEN UNPLOT 0,38
220 RETURN
```

```
  2 GOSUB 200
  5 FOR J=10 TO 1 STEP -1
 10 LET A=A+1
 11 IF A>=10 THEN CLS
 22 PRINT TAB J;"HELLO"
 30 NEXT J
 35 PRINT
 40 FOR K=1 TO 10
 50 PRINT TAB K;"ZX81"
 60 NEXT K
 65 IF A>=20 THEN GOSUB 200
 70 GOTO 5
200 LET A=0
210 CLS
220 RETURN
```

Bernard Bush, Rt. 2, Mansfield, MO 65704.

Ed, the Head

Basil Wentworth

In lines 180 to 210 an action technique is illustrated which might be useful in a number of programs, for example, a moving target, a flashing signal, alternating graphics for rotary motion.

```
 10 FAST
 20 PRINT
 30 PRINT
 40 FOR A=1 TO 3
 50 IF A=2 THEN GOTO 140
 60 FOR B=1 TO 4
 70 FOR C=1 TO 10
 80 PRINT "▒";
 90 NEXT C
100 PRINT
110 NEXT B
120 NEXT A
130 GOTO 170
140 FOR B=1 TO 2
150 FOR C=1 TO 12
160 GOTO 80
170 PRINT AT 5,7;" "
180 SLOW
190 PRINT AT 9,8;"   "
200 PRINT AT 9,8;"▒▒▒"
210 GOTO 190
```

Basil Wentworth, 1413 Elliston Dr., Bloomington, IN 47401.

perceptions

David B. Ornstein

The ZX81 Parser and User-Defined Commands

In this issue we begin a series of articles which will describe parsing, the ZX81's parser, and user-defined commands. Parsing is the process used by a computer system to extract the syntactic and semantic elements of a user's program. These elements, once isolated, are either acted upon directly or stored in a data structure (or two) for later examination. In the ZX81, and all other computers, it is the parser that is responsible for a) syntax checking, and b) execution of a program. It is one of the most important elements of any system designed to execute a language.

The Parsing Steps

When a parser is called to look at a line or a series of lines, it must perform many steps. An overview of these is as follows:

A) Let IP be a pointer to the first character to be looked-at. The character is the first one in the line to be syntax-checked/executed.

B) Let CCHAR be the character stored at location IP.

C) Process CCHAR according to the current context.

D) Increment IP.

E) If IP does not point to an End-Of-Line (EOL) character, go back to step B.

Let us now take a look at these steps. Before entry into the "SCANNING-LOOP," which ranges from steps B to E, inclusive, the parser sets up a pointer to the first character to be looked-at. This pointer, called the Interpreter Pointer, is given the label IP. Step B fetches the next character to be looked-at. The next step, C, is the most important one and will be discussed further in the next issue of *SYNC*.

Step D increments the IP. In most systems line elements are stored in contiguous memory locations, so the action performed in step D is actually an increment, i.e., IP=IP+1. To state the operation more correctly, we should say: D) Point IP to the next character to be interpreted. Step E is very straight forward. It is, conceptually, a test to see if we have looked through the whole line yet. It is implemented by a compare or a series of compares with the set of legal line-end markers.

The RST 18 and RST 20 Instructions

The ZX81's parser follows this basic outline, but it is structured a bit differently. The process used by the ZX81 to "get" characters involves two interconnected routines that are subroutines which can be called by the code responsible for line execution. These routines are labeled "GET-CURRENT-CHARACTER" and "GET-NEXT-CHARACTER." They are called by way of the RST (ReSTart) instructions: RST 18 and RST 20. The RST instruction is theoretically a CALL instruction, but it has two advantages.

The first is that it is a single byte long. The address of the routine to CALL is implicitly specified in the instruction byte itself, saving the two bytes, required by a CALL, which specify the subroutine address. With an RST, the address of the subroutine is calculated by taking the lowest three bits from the instruction and multiplying them by eight. This yields: RST 0, RST 8, RST 10, RST 18, RST 20 . . . RST 38 (all the addresses are in hex). RSTs are used for the most commonly needed/accessed routines.

Listing 1: The RST 18 and RST 20 Routines.

```
RST 0018   GET-CH:      LD  HL,(CH-ADD)     ; The "GET-CURRENT-CHAR" routine.
                        LD  A,(HL)          ; Get the char into A.
    001C   TEST-SP:     AND A               ; This instruction will test to see if
                                            ;    character in A is a space (i.e., 0).
                                            ;    It will set Z if it is a space.
                                            ;    It will set NZ if it is not.
                        RET NZ              ; Return if it is not a space.
                        NOP                 ; Padding.
                        NOP                 ; Padding.
RST 0020   NEXT-CH:     CALL CH-ADD+1       ; Increment CH-ADD.
                        JR TEST-SP          ; Go back to 001C.
```

Listing 2: The CH-ADD+1 Routine.

```
0049   CHADD+1:      LD HL, (CH-ADD)     ; the "Bump CH-ADD" routine.
004C   CURSOR-SO:    INC HL              ; bump it.
004D   TEMP-PTR:     LD (CH-ADD),HL      ; STore it.
                     LD A, (HL)          ; Get the char.
                     CP CURSOR           ; Compare it with CURSOR (7Fh).
                     RET NZ              ; If it is not a CURSOR, then return
                     JR CURSOR-SO        ; If it is, then go back, and get
                                         ;    another character to check.
```

The second advantage of using RSTs as opposed to CALLs is that the RST instruction is faster. This stems from the fact that the RST does not have to make two extra memory accesses to find out where to send the processor since this address is specified in the instruction itself.

The RST 18 and RST 20 routines are shown in Listing 1. They use an auxiliary routine called CH-ADD+1, shown in Listing 2, to move CH-ADD along to the next character. The operation performed by this routine is basically analogous to the operation performed in step D above. It is also responsible for a secondary task, namely, skipping any CURSOR characters. The CH-ADD pointer is a system variable and is used as the Interpreter Pointer in the 8K ROM.

In reviewing the routines (RST 18 and RST 20), we must note that they simply refuse to return a space character. This is why you can put spaces almost anywhere in a program without having them affect the execution of your program. The only restriction on the space-insertion rule is that all spaces inside quotation marks, i.e., valid string constants, are significant. The system facilitates this by having a separate routine to parse string constants.

These RSTs provide the base of the parser. Next time I will dive head on into the main parser routines.

Bibliography

Two bibliographic references are important for this article:

The ZX81 Monitor Listing, Part A by Ian Logan. Melbourne House, 1981.

Writing Interactive Compilers and Interpreters by P. J. Brown. New York: John Wiley & Sons, 1979.

I wish to express special thanks to Dr. Logan for writing *The ZX81 Monitor Listing*. The labels used in this article are from this work.

More on the 16K RAM Pack Schematic

I have received many calls and letters from people who want to build the RAM expansion on their own. The biggest problem with this is that some of the components listed in the power supply section are U.K. parts and are not available in the U.S. I have not been able to find cross-references for several, including the ZTX750 transistor. My suggestion to those who want to build the memory expansion is that you use the +5-volt-only version of the 4116 dynamic memory. This way the power converter section, and all the headaches that go along with it, can be eliminated.

As noted in *SYNC* 2:1, the 16K RAM pack schematic published in *SYNC* 1:5 needed some corrections and clarifications. A further correction should be made: on each 4116 (ICs 8-15) two pin 7's are shown. The one going to -5 volts should be shown as *pin number 1*.

Addenda to "The ZX80/81 Video Display System"

In the discussion of "The ZX80/81 Video Display System" in *SYNC* 2:1, one further aspect of the Display System (more specifically, the Display File) must be included. As you will note in the column, each of the 24 records in the display file is terminated by a NEWLINE (ENTER) character (76h). This is interpreted as a HALT instruction by the Z80 microprocessor. If a line in the display file is shorter than 32 characters, the system will reach and read the suffix NEWLINE.

Meanwhile, the NOP-Forcing logic has been happily forcing NOPs onto the data bus, overriding whatever happened to be coming from RAM. (Whenever the Z80 is in the middle of a display sequence and D6 from the RAM is low, the Force logic is enabled.) Whenever a character is read from the RAM, whose D6 bit is low, the NOP-Force logic is disabled, and the instruction (byte) read will be allowed to pass through to the Z80 for execution.

In the case of the record-terminating HALT instruction, this is the desired effect. The Z80 will enter a HALT loop (NOPs), not exiting from this loop until the R (Refresh) register times out and interrupts the system. Herein lies the problem with placing some special characters (those with D6 high)—usually machine code—into REM statements. If the character has the "killer-bit" set, it will pass through the NOP-Force logic and enter the Z80 for execution. This can have disastrous results.

Until next time, same relativistic time period, same non-Euclidian universe.

An Introduction to Expression Evaluation

Ian Logan

In the Sinclair 8K ROM program of the ZX80/81 the "expression evaluator" is to be found at 0F55-111Bh (improved 8K ROM) and forms an essential part of the program. A study of the fundamental parts of the "expression evaluator" can be very useful as it enables Basic programmers to understand many of the limitations and quirks of the Sinclair Basic.

Some Definitions

First we must make some definitions:

Expression. An expression is any combination of numbers, variables, strings, functions, and operators that can be combined to form the operand of a command. For example:

1) 2—a simple number.

2) 2*A—a number and a variable linked by a binary operator.

3) CHR$ 32—a function and a number.

4) CHR$(T+A-26*INT ((T+A)+38) —my favorite complex expression.

Function. Function includes the expected keywords ABS, CHR$, INT, etc., but do not ignore the two special functions NOT and -. NOT is a function that gives an "opposite logical result," i.e., NOT 3 gives 0 as the number 3 is logically "true." When - is used as a "leading minus," it gives an "opposite numeric result," i.e., -3 should be taken as"+3"subjected to the function-.

Binary operators. In the 8K ROM program the following binary operators are allowed: **, *,/, +, -, =,<=,>=,< >, AND, OR. In each case a binary operator is required to be between two operands, which may in turn be "subexpressions" in their own right.

Unary operators. There are three special operators in this group: RND, PI, and INKEY$. In one sense they are not operators at all and they can, if preferred, be considered as "fixed variables."

Dr. Ian S. Logan, 24 Nurses Lane, Skellingthorpe, Lincoln LN6 OTT, U.K.

Priority. All of the binary operators and functions have associated with them a "priority" that is used to determine the "order of the operations." Hence, 2+4*3 is 14 rather than 18 since multiplication has the priority value of 8 and addition a value of 6.

Last value. Last value is, perhaps, a difficult point to understand but the Sinclair 8K ROM program evaluates expressions to produce a "last value" on the "calculator stack." For a numeric result the "last value" is a 5 byte floating-point number and for a string result a set of 5 parameters that define the string.

The Actual "Expression Evaluator"

The object of the "evaluator" is to produce a single "last value" that corresponds to the expression that it has been given. For instance, if the expression is a simple one such as 2, then the "last value" produced will be the number 2 in binary floating-point form, and it will be the topmost number on the calculator stack. However, if the expression is 2+4*3, then the "last value" is to be 14 and this has to be "evaluated" by "scanning" the expression from left to right and saving the part-answers as they occur and the operations to be performed until the point is reached when the "saved" answers and operations have to be used.

A Simple Evaluation

A Basic line such as 10 PRINT 2 appears to be a very simple line but its interpretation by the 8K ROM program involves an almost unbelievable amount of work. Initially the interpreter has to locate line 10, then scan the line for its first command and jump to that command's routine. In this example, the jump is to the PRINT COMMAND ROUTINE at 0ACFh. In this routine a call is made to the "expression evaluator" so as to create a "last value" from the operand of the PRINT command, i.e., the 2, and then a call is made to PRINT-STK, the routine that prints the correct representation of the "last value" on the TV display. A check is then made to see if the line is finished, which in this case it is, before the next line is considered.

It can be seen in the above description that the use of the "expression evaluator" is essential and an outline view of this routine will now be given.

1) Put a zero on the machine stack as a "starting priority marker."

2) See if the first character of the expression is a "unary operator." No.

3) Is the character alphanumeric? Yes. So jump accordingly.

4) Is it a digit? Yes. So jump accordingly.

5) Transfer the invisible binary representation of the number 2 to the calculator stack as a "last value" (remember the line 10 PRINT 2 has its binary representation in 6 bytes between the 2 and the NEWLINE).

6) Fetch the next character in the line. It is a NEWLINE.

7) Jump forward to the "evaluation loop" if it is not an operator. At this point the "last operation" code is zero.

8) In the "loop" an exit is made as the "last operation" code is the same as the "starting priority marker." Both are zero.

Note the successful outcome! The "last value" on the calculator stack is the floating-point representation of 2 as was required.

The Next Stage

Let us go through this outline view of the work of the expression evaluator again, but this time the expression is 2+4*3—a seemingly innocent expression that is a great deal more complex than appears at first sight!

1) Put the "starting priority marker" of zero on the machine stack.

2) Perform the steps outlined above in 2-5 that result in a "last value" of "2" going onto the calculator stack.

3) Fetch the next character: a +.

4) This time it is an operator so prepare the "literal," 0F, and the priority of 06 for the operation of addition.

5) Enter the "evaluation loop" and as the "present priority" is greater than the "last priority" (that zero) put the "present literal" and "present priority" onto the machine stack on top of the "last literal and priority" (that zero again).

6) Now fetch the next character, the 4, and go back to step 2 again with a different operand. This time a "last value" of 4 goes on the calculator stack on top of the 2. The "literal," 04, and the corresponding priority of 08 for multiplication are prepared and the "evaluation loop" entered once again.

7) Here the "present priority" is greater than the "last priority," i.e, 08 is greater than 06, so again the "present priority and literal" go onto the machine stack. (The evaluator has determined that 2 is not to be added to 4, as the 4 is followed by a "more binding" operator.)

8) Now fetch the next character, the 3, and loop back to create a "last value" of 3, above the 4, and the 2.

9) Alas, the end of the expression has been reached. The "present operator" is zero because the present character is a NEWLINE and the "evaluator loop" is entered for the final time.

10) At this point the priorities are 00, 06, 08 and 00 whilst the calculator stack holds 2, 4, and 3. These stacked values now have to be "unstacked" and this is done as follows: The "present priority" of 00 is less than the "last priority" of 08 so the operation of multiplication, associated with the "last priority," is performed between the top two numbers of the calculator stack.

11) Now the priorities are 00, 06, and 00 and the stack holds 2 and 12. As the "present priority" 00 is less than the last priority 06, the operation of addition is performed.

12) Now the end has been reached. The priorities are 00 and 00 and, since there is but one value on the calculator stack, the required 14, the EXIT is taken.

Although the above examples have dealt only with simple decimal numbers and binary operators the other facilities of the expression evaluator are managed in a similar manner. A variable, such as A, is evaluated to a "last value" and used accordingly. A function such as COS is identified and its literal and priority prepared, and the "operation" of COSing the "last value" is performed when required. Note that it is part of the "calculator" which ensures that a unary operation replaces a "last value" with another, and a binary operation replaces two values with a "last value."

The "expression evaluator" also handles the special operators RND, PI, and INKEY$. Indeed there are no subroutines for these operators but merely segments of the "expression evaluator" deal with them in a "straight programming" manner. The addresses of the segments are (all improved ROM):

RND—0F59h
PI—0F8Ch
INKEY$—0F9Dh

The Priorities

The last point we need to cover is to correct the *ZX81 BASIC Programming* manual with regard to the "priorities" of the various operations. They are:

Operations and Functions	*Priority Decimal*
All functions (except LEADING MINUS and NOT)	16
**	10
LEADING MINUS	9
*	8
-, +	6
=, > =, <=, > , < , <>	5
NOT	4
AND	3
OR	2

Demonstration Program

Our program this time is an exercise in graphics. It is quite easy to get vehicles to go across the screen, but it is a little more complicated to get them appearing in stages from the left and disappearing slowly at the right!

Listing 1: A SYNC Delivery Van.

```
 10 SLOW
 20 LET A$="
                SYNC
                        
                              O   O
 30 FOR L=1 TO 15
 40 FOR A=37 TO 1 STEP -1
 50 PRINT AT L,0;A$(A TO A+31);
A$(A+38 TO A+69);A$(A+76 TO A+10
7)
 60 NEXT A
 70 CLS
 80 NEXT L
```

Program notes

Line 20: After the " enter 32 spaces, type in SYNC in reverse letters (hit shift and 9 to get the graphics mode and then type in the letters), enter the graphic on the R key, enter 33 spaces, enter 6 reverse spaces (get into graphics mode and hit the space key 6 times), enter 32 spaces, the letter O, 3 spaces, the letter O, and 33 spaces. Close with the ".

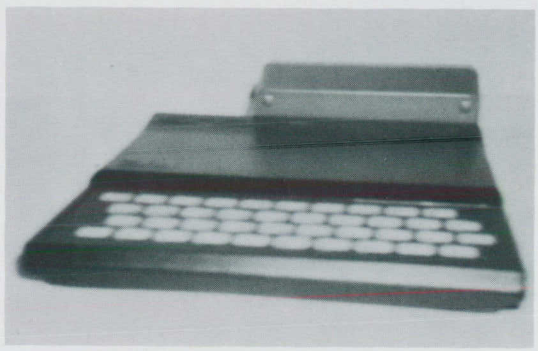
After entering the program, hit RUN and ENTER and watch the results.

Ed.—Note that Dr. Logan's **Sinclair ROM Disassembly Parts A and B** *will provide major assistance. They can be obtained from several advertisers or directly from Dr. Logan.*

ZX 81 AND TIMEX-SINCLAIR 1000 OWNERS

16K-$49.95
64K-$149.95

16K normally sells for $80, $90, or even $100. Now you can get this *vital* tool for less than *HALF* the official price. 16K transforms your ZX-81 or Timex-Sinclair 1000 into a powerful machine. Run Business, Education, and Games without running into the brick wall of "memory full".



64K is all the memory you'll ever need. In fact, its all your computer can handle. And at $149.95, this isn't just the LOWEST PRICE EVER, ANYWHERE for the ZX-81 or Timex Sinclair 1000. Its the LOWEST PRICE FOR ANY COMPUTER.

64K makes your computer more powerful than machines costing ten times as much.

These Memories are On-The-Shelf.
Just Waiting for your Order.

You bought a breakthrough when you spent your hardearned cash for a Sinclair computer. The amazing power for the rock bottom price has made the Sinclair, after only a year, the best selling computer in history. Now, you have a chance to multiply the power of that computer without spending an arm and a leg. We're so sure you're going to love the freedom and possibilities of more memory that we are GUARANTEEING THESE MEMORIES FOR 90 DAYS. Money back. No Questions asked.

Dolphin Computer
P.O. Box 3046
Redwood City, CA 94064

I NEED IT *NOW, RUSH* ME:
(I know these RAMS's are *FULLY GUARANTEED for 90 Days*)

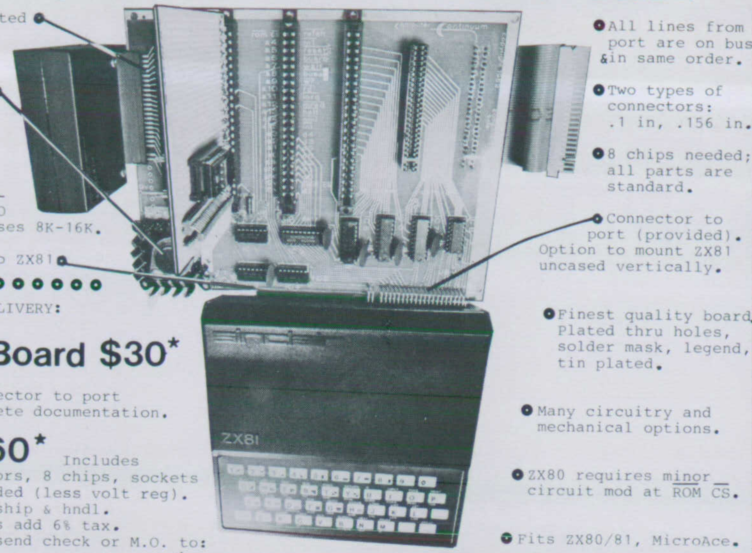
CHECK
☐ (Payable to Dolphin Computer)
MASTERCHARGE # ______
VISA# ______
SIGNATURE ______

	Quantity	Price	Total
16K RAM		$49.95	
64K RAM		$149.95	

NAME ______ POSTAGE $5.95
ADDRESS ______ TOTAL ______

OTHER PRODUCTS YOU WOULD LIKE TO SEE ______

Dolphin Computer P.O. Box 3046 Redwood City, CA 94064



try this

This column will feature short programs to show off your computer, impress your family and friends, and tickle your imagination when *SYNC* arrives at your place. We invite your contributions. Address them to: Try This, SYNC, 39 E. Hanover Ave., Morris Plains, NJ 07950.

4K ROM

Type in:
10 REM ZX80 and hit NEWLINE.
Then type in:
POKE 16431,587 and hit NEWLINE.
Hit NEWLINE again and watch the display. On some TV sets adjusting the controls will change the speed of the display.

Our thanks to:
Aaron Seeler
3460 Red Rose
Encino, CA 91436

8K ROM

ENTER the following lines:

```
 20 LET A=INT (RND*42)+1
 30 LET B=INT (RND*42)+1
 40 LET C=INT (RND*2)+1
 50 FOR J=1 TO A STEP C
 60 PLOT A,J
 70 PLOT J,A
 80 PLOT B,J
 90 PLOT J,B
100 NEXT J
110 GOTO 20
```

After you hit RUN and ENTER, watch the results.

Our thanks to:
Bernard Bush
Rt. 2
Mansfield, MO 65704

Glitchoidz Report

Getting Loaded (2:2)

Clarifications offered by the author include:
1) Check your tape recorder grounding; you may have to reverse the diode.
2) Parts list: LED #276-042; diode #276-1114.

The Game of Life Revisited (2:1)

The author has supplied the following changes which should make both versions work.

p. 20, col. 3:
LET A=USR(16427)

p. 21, Fig. 3.
1) Hex address column: the instructions after 40FD repeat Fig. 2.
2) Add the following instructions:

40AB:	CD F9 40	CALL TEST
40AE:	DD 7E 22	LD A, (IX+34)
40B1 :	CD F9 40	CALL TEST
40B4 :	DD 7E 00	LD A, (IX)

3) Correct the following instructions:

40DF:	DD 36 00 00	LD (IX),00
40E5 :	DD 36 00 80 CELL:	LD (IX),128

p. 25, Fig. 7:
4115: Correct hex format to 28 0E
Already noted in 2:2:
p. 21, Larger Field:
3) POKE 16435,20
p. 21, Fig. 5:
240 IF A<1 OR A>300...

MicroAce Video Upgrade(2:1), p. 27.

The author has supplied the following additional information:

1) 1st col., last sentence: pin 2 should be "pin 1."

2) Several readers have wondered about connection "G" on the board. The instructions can be interpreted to indicate a connection between it and both IC 21 pin 1 and the base resistor of the transistor buffer used on VHF modulator equipped ZX80 computers. Connect G as follows:

a) UHF modulators: to IC 21 pin 1.

b) VHF modulators: to the base resistor of the buffer *only*.

4K ROM 1K RAM
8KROM 16K RAM

Two Switch Human Interface for the Communicative Impaired

Charles Dorcey, Jr.

Ed.—**A WORD OF CAUTION:** *Any hardware project that involves modifications to your computer must be approached with extreme caution.* **SYNC** *cannot be responsible for problems that may arise from attempting hardware projects. Obviously, any damage done to the computer can be costly in time and money for repairs or even replacement.*

An Experiment

Let's try an experiment. Type in a few lines of the program in Figure 6, and change channels on your TV. Contemplate the result. Of course, we know that nothing is wrong with the computer; perhaps the lines are still somewhat visible. We suspect that it is still responding to its keyboard, but the responses have become "lost" somewhere along the way back to you. It is probably not worth the effort to use the computer without a readable display. Most of us would just shut it off.

Suppose now that you take the computer's point of view. You know that you are functioning as always, but that you cannot get your operator to read and respond to your output. Imagine now as a person that due to some birth defect, illness, or accident *your* messages are lost on their way out. You know that you are still as intelligent and creative as ever, but does anyone else? Do they shut you off because your message is not getting through to them?

This article is intended to use the ZX80/81 to take a step toward enhancing the communicative potential of such people. Minimum physical coordination is required to display messages on the monitor which could be placed, for example, at a bedside visible to both the user and visitor. Granted that it is crude and inefficient, but it is also quick and relatively inexpensive to build (even if you have to buy a new computer to dedicate just to this task). Hopefully, the article will stimulate a more elaborate system design.

Charles Dorcey, Jr. 912 Park Hill Rd., Laurel, MD 20707. Adapted to 8K ROM by James Grosjean.

Program Usage

The program will usually display three rows of characters on the screen at any time: the upper alphabet (UA) row, the lower alphabet (LA) row, and the message row, which will be empty at first. The program allows the user to select letters from the alphabet rows for display in the message row. Three special characters are also included: 1) the inverse-space graphic to put in a blank space; 2) the inverse less-than sign to delete the last character entered; 3) the British pound sign to delete the entire message.

Suppose the user is thirsty and wants to call for "WATER." Since "W" is found in the LA row when he begins, he pushes NEWLINE. When the display reappears, the alphabet rows are half as long. Now the UA row shows B through O; LA, P through Z. "W" is still in the LA row so he pushes NEWLINE again. The display reappears as shown in Figure 1 with UA holding P-V and LA holding W-Z. The user again chooses the lower row, and the display will look like Figure 2. Now "W" has moved to the UA row, so the user must type U and NEWLINE. Figure 3 shows the new display, and U and NEWLINE are again entered. Since there is only one letter in each row now, one more choice is needed. U and NEWLINE are entered for the third time and the "W" is put into the message line. The screen display will put the whole alphabet back for the next choice. The sequence for choosing "A" is: U, NEWLINE, NEWLINE, NEWLINE, NEWLINE, NEWLINE. Since there are 54 characters in our "alphabet," any letter can be selected with five or six choices.

Input Modifications

Since minimal control of hands and fingers often accompanies communicative disorders, some users may not be able to work with the standard ZX80/81 keyboard. For some a simple sheet of rigid material, e.g., perfboard, with finger-sized holes over the U and NEWLINE keys may be adequate. For others an external switch actuated by finger, arm, foot, or head motion will provide the necessary input control. Sound, proximity, and electromyographic sensing interfaces are beyond the scope of this article but may have potential.

Figure 1

Figure 2

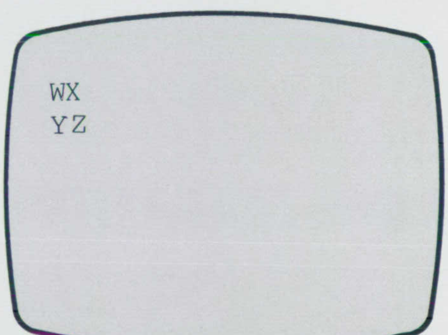
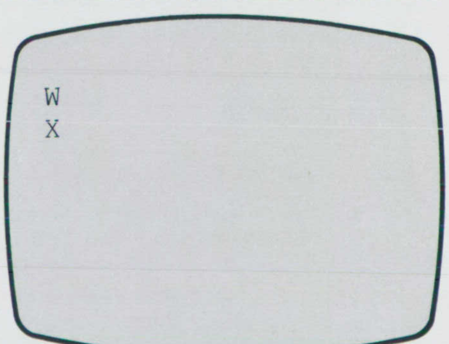


Figure 3



ZX81 **SOFTWARE** On Cassette

GAMES PACKS

1 for 1K ZX81 & 8K ROM ZX80. Eight fantastic programs for the unexpanded ZX81, including DIGICLOCK, 9-LIVES, REACTION TEST, GOBBLER and PATTERNS.
$9.95 ($12.95 in Canada)

2 for 16K ZX81. Four programs written in BASIC for the expanded ZX81. PONTOON, FRUIT MACHINE, OXO, and BIO— RHYTHMS.
$9.95 ($12.95 in Canada)

3 for 16K ZX81 and 8K ROM ZX80. Two programs for expanded ZX81 to keep you entertained for hours! 3-D OXO is written in machine code and is hard to beat. MARS RESCUE is a compulsive adventure game.
$9.95 ($12.95 in Canada)

4 for 16K ZX81. ZOMBIES — escape as they chase you around Zombie Island. Lure them into the pits, but don't fall in yourself. MOUNT MAYHEM — can you reach the 20,000 foot summit? Look out for Yetis and other hazards!
$9.95 ($12.95 in Canada)

DICTATOR

Fantastic new adventure game for 16K (or greater) ZX81. You have just become 129th ruler of Ritimba with a single goal in mind: take full advantage of the situation for your own good. You have to deal with a handful of factions: unruly army, downtrodden peasants — but you have the secret police on your side.
$14.95 ($17.95 in Canada)

CONSTELLATION

Turn your ZX81 into a telescope with this amazing 16K program. Produces a simulation of the night sky as seen from any position on Earth at any chosen time this century. You may point your telescope in any direction, move it up, down, left or right, zoom in or out. Stars may be displayed by magnitude or constellation
$14.95 ($19.95 in Canada)

CHESS

A challenging chess programme, written in machine language, designed to operate in the ZX81 fast mode. ZX Chess allows you to select from 6 levels of play, choose either black or white, and enables castling and en passant moves. Unique "self-running" feature: you start the tape and when the chess board appears on the screen, start your game.
ZX CHESS! **$24.95** ($29.95 in Canada)

ZXAS Assembler

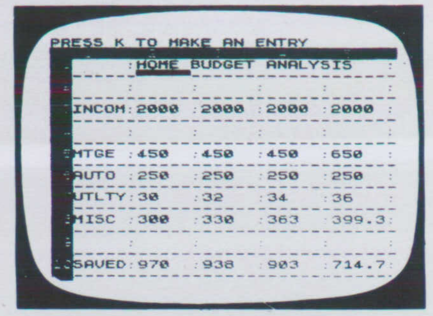
Now you can use the full power of the Z80 microprocessor without having to laboriously POKE in instruction codes. This full specification Z80 assembler assembles all the standard Zilog mnemonics, which are simply written into REM statements (more than one per line is allowed) within your BASIC program. When assembled, the assembly listings, together with assembled codes and adresses, are displayed on the screen. The assembled code is executed by USR. The program occupies 5K, is situated at the top of the memory, and is protected from overwriting. This means that ZXAS may be used in conjunction with ZXDB (see below), providing an extremely powerful machine code system normally only found on very expensive computers.
The program is available for both the ZX81 and the 8K ROM ZX80, and in both cases, the 16K RAM pack is required. Despite the low price, ZXAS is a FULL-SPECIFICATION assembler, and is a must for all serious ZX users. Full documentation on how to use the assembler (including a list of the mnemonics) is supplied.
$9.95 ($12.95 in Canada)

ZXDB Disassembler/ Debugger

The perfect complement to the ZXAS assembler, ZXDB is a complete combined machine code disassembler and debugging program. Like ZXAS, it is itself written in machine code for compactness, and may be used in conjunction with ZXAS, still leaving about 9K of memory for your own program.
Apart from the DIASSEMBLER, the program has features including SINGLE STEP, BLOCK SEARCH, TRANSFER AND FILL, HEX LOADER, REGISTER DISPLAY and more, all of which are executed by simple one key commands from the keyboard. All in all, an extremely powerful programming aid, well worth the money for the disassembler alone!
$9.95 ($12.95 in Canada)

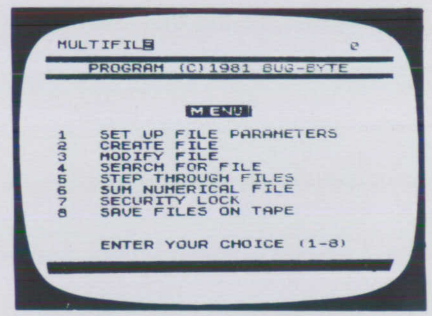
STAR TREK

The classic computer game in which you trek across the galaxy in search of Klingons to zap with your phasers and photon torpedoes. You have long and short range scanners to help you find them, Starbases to refuel your ship at and, of course, various witty comments from the crew. 16K.
$9.95 ($12.95 in Canada)



VU-CALC

VU-CALC. Constructs, generates, and calculates large tables for analysis, budget sheets and projections. Up to 26 columns of figures or data can be entered, plus user definable formulae capable of relating any one or more position in the table to any other defined position.
$24.95 ($29.95 in Canada)



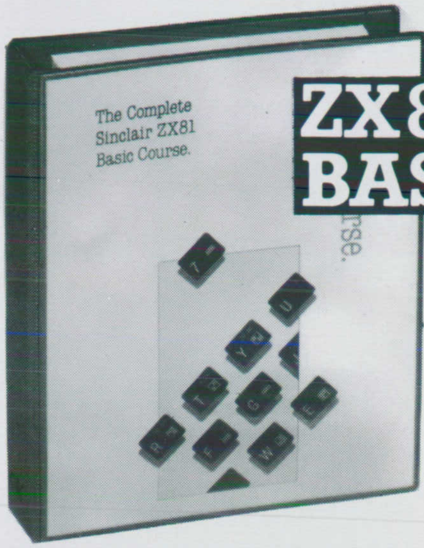
MULTIFILE

Data Storage System
An amazingly versatile multi-purpose filing system for the 16K ZX81. The program is menu-driven, and number, size and headings of files are user-definable. Both string and numerical files are catered for. Files may be created, modified, replaced, and searched, and are protected by an ingenious foolproof security system. Output to the ZX printer is also provided. The program comes on cassette, together with three quality data cassettes for file storage, and comprehensive documentation, describing a host of applications for both business and personal use. Supplied in an attractive storage case. If your ZX81 is bored with playing games, then this program will give it plenty to think about!
$29.95 ($39.95 in Canada)

VIEWTEXT

A ten page information display system for the 16K ZX81. Can display both text and graphics in any sequence with variable speed. Many applications including shop window displays, educations, animation, etc.
$14.95 ($17.95 in Canada)

Gladstone Electronics, 901 Furhmann Blvd., Buffalo, N.Y. 14203 (716) 849-0735



ZX81 COMPLETE BASIC COURSE

The Complete BASIC Course contains over 100 programs and examples! These programs illustrate the use and possibilities of the Sinclair ZX 81:

- Home use
- Financial analysis and planning
- Educational applications
- Games
- Mathematical applications
- Displays of Artificial Intelligence

EVERY FUNCTION COVERED:

No matter what your application, what your confusion about any function, you will find it covered in the Complete BASIC Course.

A full and detailed discussion is included of even traditionally taboo topics such as USR, PEEK and POKE.

A handy alphabetical summary section lists all functions, and provides a short description and example programs of all topics.

A PERMANENT WORK OF REFERENCE:

The Complete BASIC Course is an excellent reference work for experienced programmers (including tips on using special techniques) as well as a comprehensive step-by-step guide for complete beginners.

The Complete BASIC Course has over 240 pages filled with information in an attractive durable ring binder – this is a lay-flat work of reference that deserves a place next to every Sinclair ZX 81 microcomputer.

$34.95

At last, a comprehensive text for your Sinclair ZX 81! The complete BASIC Course is a manual which will immediately become an indispensible work of reference for all your ZX 81 programming.

Whether you have never done any programming or whether you are an experienced microcomputer user, the Complete BASIC Course will provide itself to you as an invaluable aid.

The Complete BASIC Course is designed to teach you to write and develop BASIC programs for the Sinclair ZX 81 – no other books or aids are necessary. All is revealed in our easy step-by-step guide with programs and "test yourself" exercises all the way through.

As you become more proficient with computing, the Complete BASIC Course will continue to be an essential guide, giving you finger tip references, numerous advanced programming techniques and memory saving devices specifically for the Sinclair ZX 81.

HOW TO WRITE PROGRAMS:

Even if the idea of writing programs is completely mystifying to you, the Complete BASIC Course will show you just how easy it is. In no time you will be able to write and enjoy complex programs for whatever use you desire.

Using the proven "TOP-DOWN" approach, the Complete BASIC Course will show you systematic and simple ways to write programs. Even experienced programmers will benefit from this Course, making programs easier to write and less prone to error!

NUMEROUS EXAMPLES:

Every concept, every function is fully described by simple programs that you can enter on your Sinclair ZX 81 in minutes.

MORE BOOKS!

Not Only 30 Programs for the Sinclair ZX81: 1K-
Not only over 30 programs, from arcade games to the final challenging Draughts playing program, which all fit into the unexpahded 1K Sinclair ZX81 but also notes on how these programs were written and special tips! Great value!
$14.95 ($16.95 in Canada)

Machine Language Programming Made Simple for the Sinclair
A complete beginner's guide to the computer's own language—Z80 machine language. Machine language programs enable you to save on memory and typically give you programs than run 10-30 times faster than BASIC programs.
$19.95 ($23.50 in Canada)

Understanding Your ZX81 ROM
A brilliant guide for more experienced programmers by Dr. Ian Logan, this book illustrates the Sinclair's own operating system and how you can use it. Includes special section on how to use machine code routines in your BASIC programs.
$19.95 ($23.95 in Canada)

ZX81 ROM Disassembly
Part A. Covers ROM locations from 000H to OF54H and includes all functions except for routines used in floating point calculator. A must for the experienced programmer.
$14.95 ($16.95 in Canada)

BASIC Course Programs on Cassette
All major programs in the BASIC Course are available pre-recorded in this set of cassettes. This is a valuable adjunct to the Course, saving you time and effort.
$7.50 ($9.95 in Canada)

LAST MINUTE SOFTWARE!

Backgammon (16K) Includes a high quality graphic representation of the board. **$14.95** ($17.95 in Canada)
Flight Simulation Incredible realistic machine code program for piloting your own ZX81! **$24.95** ($29.95 in Canada)
Damsel & the Beast (1K) Superb adventure game. **$14.95** ($17.95 in Canada)
Two Adventure Games (16K) **$14.95** ($17.95 in Canada)

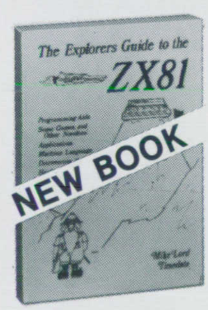
GLADSTONE
Electronics 901 Fuhrmann Blvd., Buffalo, NY, 14203

64K For Sinclair ZX81
$179-95

Memotech 64K Memopak

The Memopak is a 64K RAM pack which extends the memory of the ZX81 by a further 56K. Designed to be in the price range expected by Sinclair owners. Plugs directly into the back of the ZX81 and does not inhibit the use of the printer or other add-on boards. There is no need for additional power supply or cables. The Memopack together with the ZX81 gives a full 64K, which is neither switched nor paged, and is directly addressable. The unit is user transparent, and accepts such basic commands such as 10DIM A(9000). With the Memopak extension the ZX81 is transformed into a powerful computer, suitable for business, leisure and educational use, at a fraction of the cost of comparable systems.

The Explorers' Guide to the ZX81



Programming Aids
Some Games and Other Novelties
Applications
Machine Language
Discovering the ROM
Hardware
$9.95 ($12.95 in Canada)

Shipping and handling charge $1.50 per order books and cassettes, $3.00 per order 64k Memopak.
In Canada: **Gladstone Electronics**, 1736 Avenue Rd., Toronto, Ont. (416) 787-1448.

CHARGE TO:
☐ VISA ☐ MASTERCARD
account number:

Expiry date

Signature

Date

Please rush me:

Please rush me:	Quantity	Price	Total

NAME

ADDRESS

SHIPPING

TOTAL

To: Gladstone Electronics, 901 Furhmann Blvd., Buffalo, N.Y. 14203 (716) 849-0735

Figure 4. Add on Switch Wiring

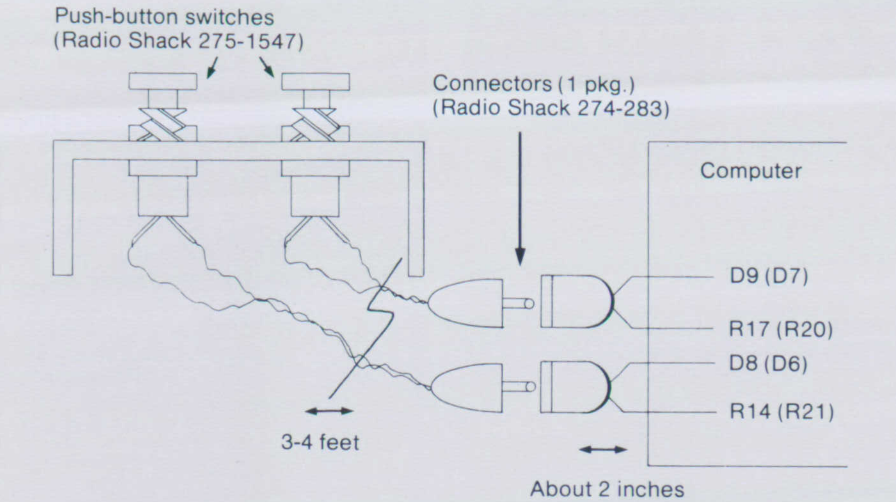
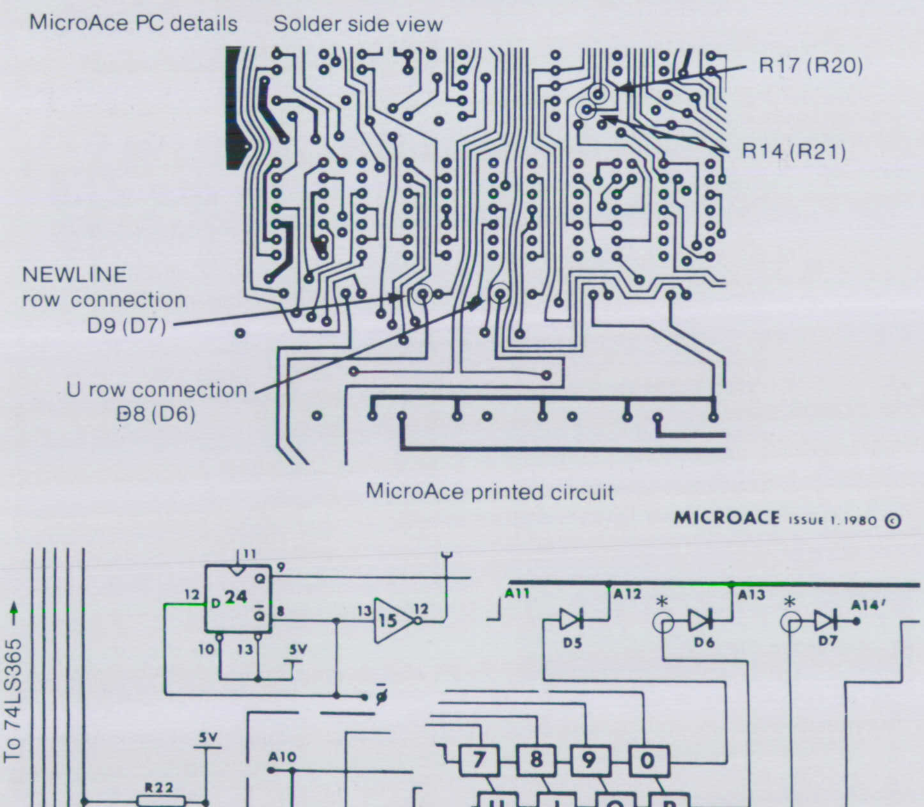


Figure 5. PC Board Detail



A Hardware Modification

Adding the necessary switches to the computer should be no problem for anyone who knows which end of a soldering iron to hold. In this discussion specific references to the ZX80 will be followed by the corresponding MicroAce numbers in parentheses. For example, on both schematic diagrams, five lines come down from the keyboard matrix, go left across the bottom and up the left side into a 74LS365 integrated circuit. This IC is designated IC10 (U11) on the pc board and schematic. I have not had a ZX81 to work with, but since the keyboard is electronically identical only the part numbers would change.

It is helpful, though not essential, to understand how the original ZX80 keyboard works. Under each key is a square in which two wires cross to form a simple pair of contacts which come together when the key position is pressed. Though the keyboard is physically arranged as four rows of ten keys, electrically it is arranged as eight rows of five columns. Each column is wired to a bit on a data bus which goes up through that 74LS365 to the Z80's data bus when the processor is testing the keyboard. Since the column wires are "pulled up" to 5v ordinarily, the only time the Z80 will see a zero bit when testing the keyboard is when one of those keyboard "switches" connects the column line to a row line which is grounded. A short circuit to ground "pulls" down harder than a resistor can pull up. By rapidly grounding one row line after another, the CPU can test all the rows in the keyboard. But how does it do that? A very subtle feature, probably unique to the Z80, is involved. The monitor program executes an "IN C,kbd" instruction which 1) puts out the keyboard address on the low half of the data bus, 2) loads the current pattern from the data bus into the C register of the CPU, and 3) (this is the sneaky part) drives the high half of the address bus with the contents of the CPU's A register. This high half of the address bus drives the column lines of your keyboard just the right way so one column is sensed at a time and nobody gets confused but the poor hobbyist who tries to figure the whole thing out.

Two wires must be added to your computer for each extension switch. All the switch has to do is short the right row to the right column to imitate the original switch. The type of switches to use will vary with the intended user's capabilities, but should be normally-open (N.O.), single pole single throw ((SPST), either push-buttons or spring-return toggles. To allow easy removal of the switches when not in use, add a pair of audio-type miniature connectors in the switch wiring, but that is optional. Proceed as follows:

1) Run one wire from D9 (D7) to one terminal of a switch. Use the end of the diode which is closest to the keyboard wiring. See Figure 4.

2) Run another wire from R17 (R20) to the other terminal of the same switch. Use the end of the resistor which is NOT connected in common with the others to +5v. This is the NEWLINE extension switch.

3) Add wires to D8 (D6) and R14 (R21), as described in 1) and 2) above, for the "U" switch.

4) You are now ready to enter and RUN the program.

If you do your own soldering, use as fine a tip as you can get. Keep it clean and bright. When the iron is dirty (dull grey in color), solder "bridges" to adjacent conductors are easy to make and hard to remove. Use no more solder than necessary.

Figure 6:
Basic Output Enhancement I, 4K Program

```
  10 REM BASIC OUTPUT ENHANCEMEN
T I
  20 DIM A(53)
  30 DIM T(64)
  40 FOR I=0 TO 51
  50 LET A(I)=I+12
  60 NEXT I
  70 LET A(52)=128
  80 LET A(53)=152
  90 FOR J=1 TO 63
 100 LET A=0
 110 LET B=26
 120 LET C=53
 130 CLS
 140 IF J<63 THEN GOTO 160
 150 PRINT "DISPLAY FULL"
 160 FOR I=A TO B
 170 PRINT CHR$ (A(I));
 180 NEXT I
 190 PRINT
 200 FOR I=B+1 TO C
 210 PRINT CHR$ (A(I));
 220 NEXT I
 230 PRINT
 240 PRINT
 250 FOR I=1 TO J-1
 260 PRINT CHR$ (T(I));
 270 NEXT I
 280 PRINT
 290 INPUT A$
 300 IF A$="Q" THEN STOP
 310 IF CODE (A$)=58 THEN GOTO 3
50
 320 LET A=B+1
 330 LET B=(A+C)/2
 340 GOTO 370
 350 LET C=B
 360 LET B=(B+A)/2
 370 IF A<B OR B<C THEN GOTO 130
 380 LET T(J)=A(C)
 390 IF A(C)=152 THEN LET J=J-2
 400 IF A(C)=12 THEN GOTO 90
 410 IF A(C)=128 THEN LET T(J)=0
 420 NEXT J
 430 GOTO 90
```

Figure 7:
Basic Output Enhancement I, 8K Program

```
  20 DIM A(54)
  30 DIM T(65)
  40 FOR I=1 TO 52
  50 LET A(I)=I+11
  60 NEXT I
  70 LET A(53)=128
  80 LET A(54)=147
  90 FOR J=1 TO 64
 100 LET A=1
 110 LET B=26
 120 LET C=53
 130 CLS
 140 IF J<64 THEN GOTO 160
 150 PRINT "DISPLAY FULL"
 160 FOR I=A TO B
 170 PRINT CHR$ (A(I));
 180 NEXT I
 190 PRINT
 200 FOR I=B+1 TO C
 210 PRINT CHR$ (A(I));
 220 NEXT I
 230 PRINT
 240 PRINT
 250 FOR I=1 TO J-1
 260 PRINT CHR$ (T(I));
 270 NEXT I
 280 PRINT
 290 INPUT A$
 300 IF A$="Q" THEN STOP
 310 IF A$="U" THEN GOTO 350
 320 LET A=B+1
 330 LET B=INT ((A+C)/2)
 340 GOTO 370
 350 LET C=B
 360 LET B=INT ((B+A)/2)
 370 IF A<B OR B<C THEN GOTO 130
 380 LET T(J)=A(C)
 390 IF A(C)=147 THEN LET J=J-2
 400 IF A(C)=12 THEN GOTO 90
 410 IF A(C)=128 THEN LET T(J)=0
 420 NEXT J
 430 GOTO 90
```

Suggestions for Program Improvements

Obviously, improvements can be made to the program in Figure 6. Rewriting it in machine language would speed it up and remove the need to hit NEWLINE each time. A user defined list of words either in place of or in addition to the alphabet would remove the laborious retyping of common words and phrases. If you want to get fancier, alternate displaying the top and bottom rows so that a single switch closure during the appropriate display period would then do the selection. (One sound-activated switch is much easier to build than two.) Finally, a "maximum-entropy" coding scheme could also speed use by creating shorter "paths" to commonly needed symbols (whether individual characters, letter groups, or words). If you put only the four most common symbols in the LA row in the beginning, only three decisions would be needed to select any one of them. (Note: this is getting dangerously close to "information theory.")

Program Notes

1) The program was written on a 4K ROM, 1K RAM MicroAce.

2) The A array holds the "alphabet." The symbol set (lines 50-90) can be customized for the individual.

3) The T array (64 letters) holds the message text. Modify lines 30, 90, and 140 to expand it for your memory capacity.

4) Line 300 provides for modification and/or debugging while running the program by entering "Q" to quit.

5) Any input line beginning with U selects the UA row. Otherwise the LA row is selected. When only one character remains in the selected row, it is entered into the text buffer, i.e., the message line.

6) Lines 390, 400, and 410 select the control symbols and can be changed to your own preference. The [£] resets the file pointer and, in effect, wipes the slate clean. The ◀ moves the pointer back one space to allow error correction. The ■ puts a space into the text buffer. ▚

Keyboard/System Conversion: The First 40/1 Keys

Robert B. Trelease

Ed. — **A WORD OF CAUTION:** *Any hardware project that involves modifications to your computer must be approached with extreme caution.* **SYNC** *cannot be responsible for problems that may arise from attempting hardware projects. Obviously, any damage done to the computer can be costly in time and money for repairs or even replacement.*

Introduction

As a daily user of some of Digital Equipment Corporation's nicest micro-based laboratory computer systems, it is easy for me to be prejudiced about system design and operating convenience. In particular, I will have to admit a real fondness for DEC's VT-103/LSI-11 ergonomically designed video terminal with detached keyboard and self-contained 16 bit LSI-11 microcomputer bus/backplane.

As a ZX80 owner, on the other hand, I also find system simplicity and portability very attractive. As others have pointed out, one only needs a battery pack, a small portable television, and a ZX80/1 or MicroAce for a backpackable combination that can bring computing to places like the high Sierra Nevada wilderness.

On first thought, it would seem a considerable design problem to configure a system combining DEC-style keyboard convenience and modularity with Sinclair-style portability. As it turns out, the problem is not very great, and the average (read: technically sophisticated) *SYNC* reader can probably handle it with ease.

The purpose of this article is to review computer keyboard (KB) conversion for the ZX80 or MicroAce and to describe the first phase of an adaptation that retains original system portability while allowing upgrading to an "advanced" expansion-bus oriented system.

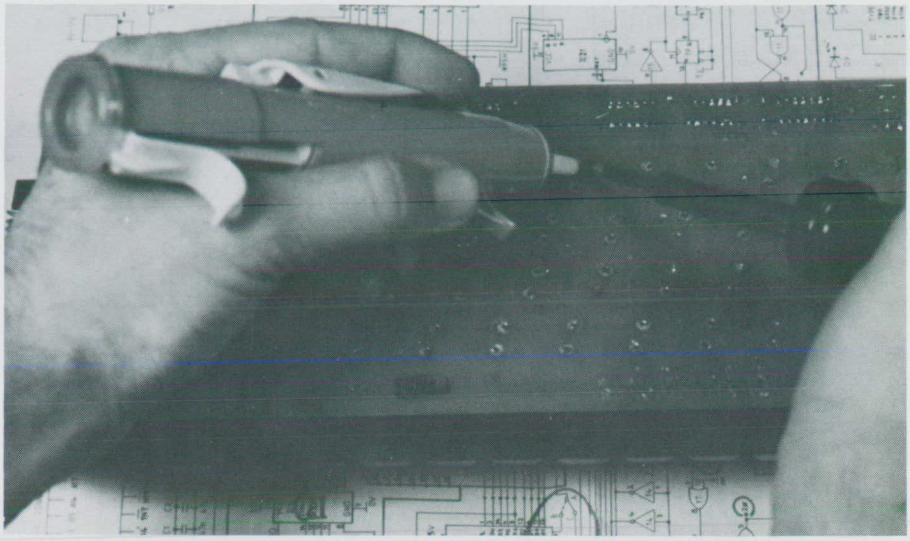
Keyboard Conversion Kits

The fundamental conversion described here can be made with almost any KB switch array, 40 keys, new or used, so long as the switches are normally open and are closed with a keystroke. Such

Robert B. Trelease, 2313 5th St., Santa Monica, CA 90405

keyboards can then be paralleled with the ZX80 keyboard. The Schultz Systems keyboard used in this article was chosen because the kit included the electronics for additional features not available with the standard Sinclair or MicroAce. These features, which include shift-locking, auto-repeat, and single key, auto-upshifted functions (such as single key rubout) will be covered in a future article on advanced conversions. Unfortunately, Schultz no longer provides complete conversion kits with used KB's, although complete instructions, KB source listing, and other parts can still be obtained. (At this writing kits and instructions are available from L.J.H. Enterprises and Double H Electronics.)

Figure 1: Desoldering the keyboard.



Construction Details

The KB provided, as is probably the case with most used units, came equipped with a printed circuit board electronics matrix attached to the switch terminals. To begin the conversion, it was thus necessary to remove this PC board. A de-soldering tool, such as a SOLDAVAC (TM) (Radio Shack no. 64-2085 or equivalent) is a real necessity. Even with this tool, unsoldering the old matrix board was clearly the most difficult part of the initial conversion, taking about 2 of the 4 hours needed for the project. In order to avoid switch damage on removing the PC board, all of the solder had to be removed from each connection. This required 2 complete passes over the board. (Figure 1)

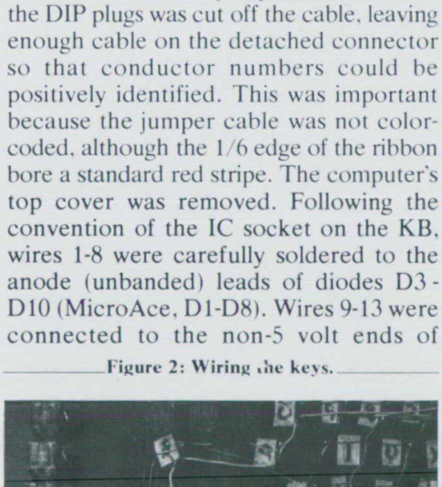
Once removed, the old matrix board was kept for salvageable parts like sockets, IC's, diodes, resistors, and capacitors—an added bonus! Each switch was tested with an ohmmeter in order to avoid unpleasant surprises and then labelled on the back to aid in wiring. Although it was not necessary with my KB, the circuit board could have been retained for switch support, with each terminal contact being isolated from the old circuit by cutting printed circuit conductors.

The switches were then strung together in rows and columns using the ZX80 matrix convention (see *SYNC* 3:42 for greater detail). Briefly, this consists of wiring in parallel one terminal of each switch in 8 rows (e.g., "shift," Z, X, C, V) and the other terminal of each switch in 5 "folded" columns 9 to 13. Since there were two "shift" keys, each was wired the same to allow shifting with either hand. For convenience, I did the same with adjacent "," and "." keys. All connections were made with standard insulated hookup wire. (Figure 2)

For connecting the wired KB matrix to the computer, I used a 16 pin DIP terminated 18" ribbon "jumper" cable (Radio Shack 276-1976) and 2 16 pin DIP IC sockets (Radio Shack 276-1998). A small piece of 0.10 spacing "perf-board" was attached to a convenient slot in the KB frame using cyanoacrylate glue ("Crazy glue"). One of the 16 pin IC sockets was glued to the board, and the row/column leads (1-8/9-13) were soldered to its terminals in numerical order (1-8 and 9-13 respectively). Pins 14-16 were reserved for power connections for the advanced functions. (Figure 3)

The final step of the basic conversion, connection to the computer board matrix diodes and resistors, was carried out using the DIP terminated jumper cable. One of the DIP plugs was cut off the cable, leaving enough cable on the detached connector so that conductor numbers could be positively identified. This was important because the jumper cable was not color-coded, although the 1/6 edge of the ribbon bore a standard red stripe. The computer's top cover was removed. Following the convention of the IC socket on the KB, wires 1-8 were carefully soldered to the anode (unbanded) leads of diodes D3 - D10 (MicroAce, D1-D8). Wires 9-13 were connected to the non-5 volt ends of resistors R13-R17 (MicroAce R18-R22; neither system is in numerical order!). To avoid component damage, wire leads were pretinned, and contacts were only briefly heated (Figure 4)

Figure 2: Wiring the keys.

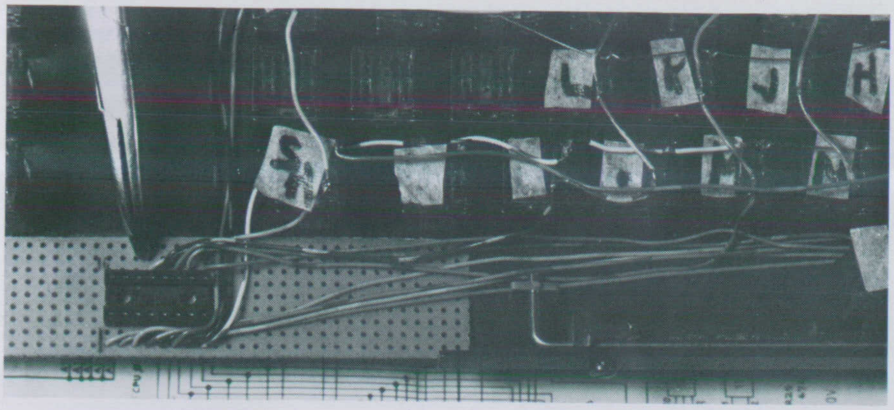


In my ZX80, the ribbon cable was led back under the voltage regulator heat sink to exit the case just above the expansion bus. (Other exits could be used in the ZX80 or MicroAce. For example, a small slot might be cut in the cover). Strain relief was provided by tethering the ribbon cable to the video modulator case. (Figure 5)

The final products of this conversion were a DIP-socketed, wired keyboard and a ZX80 with a mating ribbon cable "tail." When the system was connected, everything worked—almost. Initially, the R and O keys did not function on the new keyboard. Closer examination showed that in wiring the matrix, I had slightly bent one terminal on each key so the switch contacts did not close. A gentle push on the terminal connection restored complete function. I then had a ZX80 with both keyboards functioning in parallel. (Figure 6)

In order to ruggedize the project and reduce TV interference from the conversion, a keyboard enclosure was added. I chose an aluminum housing for my system (BUD KB 13202 from Herbach and Rademan, Inc.; depending on the specific KB adapted, many other enclosures could be used). The IC socket on the keyboard was jumpered to another socket mounted on the back of the enclosure. (For courageous engineers who do not wish to test keyboard wiring before enclosing, connections from the rows and columns may be made directly to an IC socket on the cabinet). The connector pins on the computer's new "tail" were protected during "stand-alone" operation using the remaining 16 pin IC socket.

Figure 3: Attaching row/column leads to IC socket on perfboard.



Since the keyboard was printed with the standard type letters as well as with unusual "uppercase" functions, a copy of the ZX80 template was used as a "cheat-sheet." Transfer lettering and clear epoxy lacquer also could have been used to identify keys.

Synopsis and Provision for Expansion

As described here, the basic keyboard conversion is easy to wire, taking a slow builder about four hours of straight time. The finished product as modified is a detachable keyboard for the ZX80/1 or MicroAce which can serve as the foundation for a more complex, expansion bus oriented microcomputer system. In the present form, the original system can be separated from the new keyboard, thus retaining true portability.

By implementing CMOS switching circuitry such as that obtainable from Schultz Systems, one can obtain advanced keyboard functions like auto-repeat and single-key rubout. By adding a separate cabinet containing a heavy duty power supply, printed circuit card guides, and ZX80 bus connectors, the true "hardware freak" can inexpensively construct a modular microcomputer system with features similar to those of more expensive business and scientific models. (A somewhat different approach to such an advanced system was shown in *SYNC* 4:38 "A Parallel Interface" by Alger Salt.) Design and implementation of these advanced features will be considered in a future article.

(Note: DEC, VT-103, and LSI-11 are trademarks of the Digital Equipment Corporation.)

ZX81 Conversion Details

Shortly after finishing the first part of this conversion article, I had the good fortune to acquire a ZX81 kit. It immediately became apparent that the same general conversion scheme could be applied to the new machine as well. I will detail only one approach, although *SYNC* readers will undoubtedly conceive of other ways to do the job.

The keyboard matrix and connector should be constructed as detailed above, with rows and columns wired to 13 pins on the attached DIP socket. The jumper cable should then be connected to the matrix diodes and resistors, which are D1-D8 and RP-3 in the ZX81. In the quick and dirty approach, the cable leads should be conservatively stripped, pretinned, and then soldered to the PC traces of connectors KB 1 and KB 2 on the solder side of the board.

Figure 4: Attaching ribbon wires to the diodes.

Soldering may be done relatively easily with only the bottom half of the computer case removed. Figure 7 depicts the PCB under KB 1 and KB 2 and shows the order of connections for the lines of the jumper cable.

Stripping of the jumper leads should be kept to a minimum, and some additional insulation, like heat shrink tubing or silicone cement, may be used to reduce the likelihood of shorting. The cable may be out of the case below the bus connector, and tethering to PCB screw channel/support posts will help reduce the chances of wire breaks and shorts.

ZX81 owners particular about the "OEM" condition of their machines might want to try adapting a 16 pin dip plug to flexible printed circuit "cable" for direct connection to KB 1 and KB 2. This scheme necessitates opening the case, however, and disconnecting the membrane KB when connecting to the new KB. This is clearly no disadvantage for those who might wish to mount the ZX81 or bare PCB permanently inside the new KB enclosure.

Figure 5: Ribbon cable exit from computer.

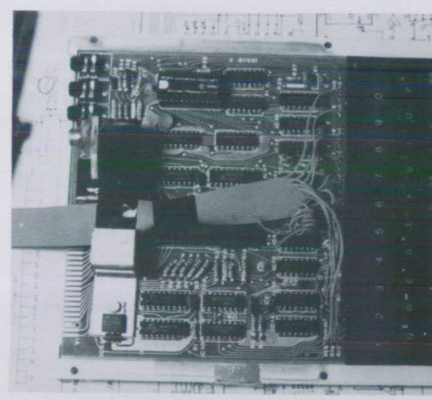


Figure 6: Ribbon cable attached to IC socket.

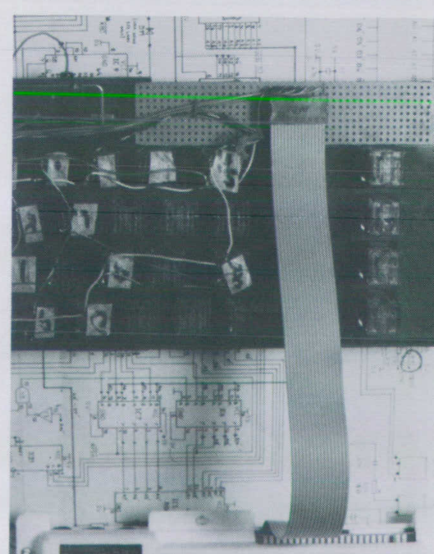


Figure 7: ZX81 hookup.

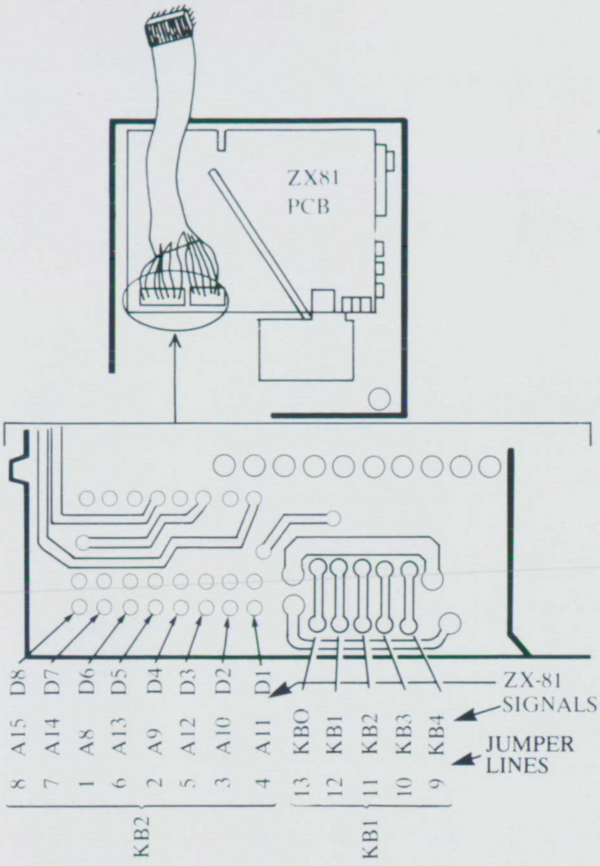
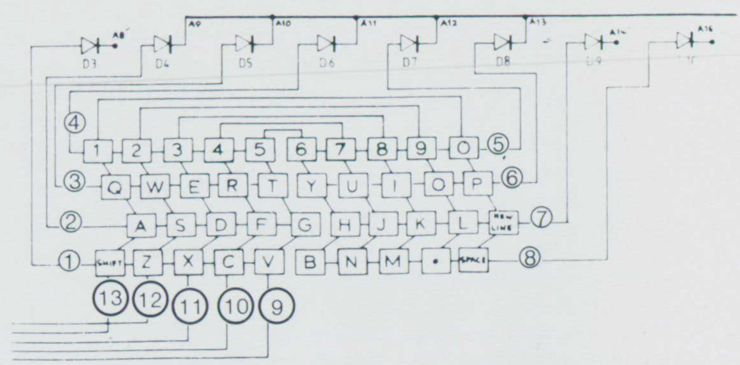
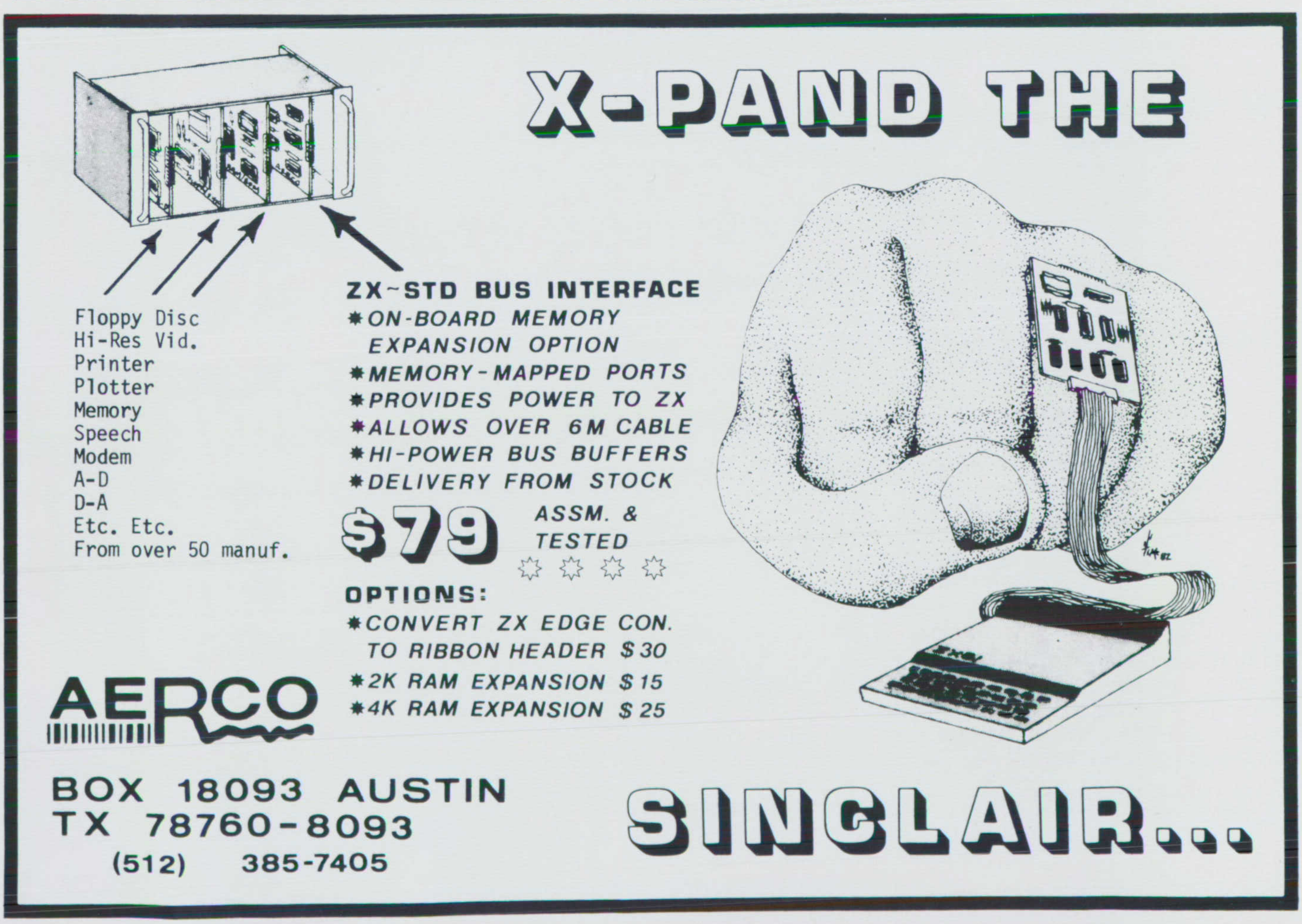


Figure 8: Row and column arrangement.





8K ROM
16K RAM

Least Squares Data Analysis with the ZX80/81

Alger Salt

One of the most common tasks of a computer is to perform statistical analysis and data manipulative type computations. Many of the computations involved are very tedious to perform by hand since they sometimes require a large number of reiterative calculations. Statistical analysis problems that take hours for a human (equipped only with pencil and paper) are solved in a matter of seconds with a computer. This article presents a program which computes some simple statistical quantities commonly used by engineers and scientists.

Data collected in the laboratory or in the field often follows some logical pattern. The quantity of interest depends on or is a function of another quantity. By recording a sufficient amount of data, this relationship may be determined by fitting a function to the set of data points. It is rare that data collected in the real world corresponds exactly to some function; rather it is scattered about with a certain probability of corresponding to the function.

Consider a function which would describe the ambient temperature in Cove City, North Carolina, as a function of time for the last twenty years. If the function were to include all of the daily temperature fluctuations, it would indeed be a very complicated function. However, if only the average weekly or monthly temperatures were recorded, then a reasonable function could be fitted to the finite number of data points. The function would resemble a sinosoid with a period of 365 days with maxima and minima occurring in the late summer and winter respectively. There would be some uncertainty associated with correlating a particular day in the future with some specific temperature. In this case the uncertainty lies within some confidence limits determined by the climate. In other cases the uncertainty depends primarily on the method or apparatus used to measure the data.

Fitting a function to a set of data points is desirable because it allows one to make predictions or extrapolations from hypothetical or trial data.

The Method of Least Squares

In many situations the function which relates one quantity to another is linear. That is, if the data are plotted, the points lie on or about straight line. Examples include: the current through a resistor as a function of voltage, the conductance of pure water as a function of the amount of salt added or the length of a suspended spring as a function of the load attached. An approximate method of determining the "best" straight line through a set of data points is by plotting the points on graph paper and drawing a line which uniformly divides all the points. Of course, this method is prone to error because determining the "best" straight line is somewhat subjective and two persons working with the same data will rarely arrive at identical functions.

The "best" straight line through a set of data points is defined as that line in which the sum of the squares of the deviations of all points from the line is a minimum. This method of "least squares" is far superior because it calculates, not approximates, the line. The method yields two quantities: the slope and the intercept of the line, thereby defining the linear function by the well-known relation

$$y = ax + b$$

where a is the slope and b is the y-intercept.

The slope and intercept are determined by the formulas in Figure 1. Another useful quantity is the correlation coefficient, given in Figure 2, which is the relative amount

Figure 1.

$$\text{slope} = a = \frac{xy - \frac{x\ y}{N}}{x^2 \quad \frac{(\ x)^2}{N}}$$

$$\text{y-intercept} = b = \frac{y - m\ x}{N}$$

Figure 2.

$$\text{correlation coefficient} = c = m \quad \frac{\dfrac{x^2 - \frac{(\ x)^2}{N}}{N-1}}{\dfrac{y^2 - \frac{(\ y)^2}{N}}{N-1}}$$

Alger Salt, Department of Chemistry, East Carolina University, Greenville, NC 27834.

NEW SUBSCRIBERS ONLY

Send me Sync at up to 33% off!

- ☐ I want one year (6 issues) of *Sync*, the magazine for Sinclair users, for only $12.97—19% off!
- ☐ I prefer two years (12 issues) for $22.97—28% off!
- ☐ Make that three years (18 issues) for $31.97—33% off!

Savings based on full one-year subscription price of $16.

Mr./Mrs./Ms. ______________________
(please print full name) 4S03

Address ______________________ Apt. ______

City ______________________

State ______________ Zip ______________

CHECK ONE: ☐ Payment enclosed. ☐ Bill me.

Foreign postage: Add $3 a year for Canada. Add $5 a year (cash payment in U.S. currency only) for other countries outside U.S. and possessions. Please allow 60 to 90 days for delivery of first issue.

SY-C4-1000

NEW SUBSCRIBERS ONLY

Send me Sync at up to 33% off!

- ☐ I want one year (6 issues) of *Sync*, the magazine for Sinclair users, for only $12.97—19% off!
- ☐ I prefer two years (12 issues) for $22.97—28% off!
- ☐ Make that three years (18 issues) for $31.97—33% off!

Savings based on full one-year subscription price of $16.

Mr./Mrs./Ms. ______________________
(please print full name) 4S03

Address ______________________ Apt. ______

City ______________________

State ______________ Zip ______________

CHECK ONE: ☐ Payment enclosed. ☐ Bill me.

Foreign postage: Add $3 a year for Canada. Add $5 a year (cash payment in U.S. currency only) for other countries outside U.S. and possessions. Please allow 60 to 90 days for delivery of first issue.

SY-C4-1000

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 738 MORRISTOWN, NJ

POSTAGE WILL BE PAID BY ADDRESSEE

The magazine for Sinclair users

SYNC

CN 1986
Morristown, New Jersey 07960

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 738 MORRISTOWN, NJ

POSTAGE WILL BE PAID BY ADDRESSEE

The magazine for Sinclair users

SYNC

CN 1986
Morristown, New Jersey 07960

of scatter about the line. A correlation coefficient of 1 means that all data points lie exactly on the least squares line. A correlation coefficient of less than 0.9 implies that the data are either invalid or are not linearly related.

The Program

Though the program as written will not fit into a 1K system, it can easily be broken down into independent modules so that 1K owners should still be able to benefit from this article. The DIM statements at lines 10 and 11 limit the maximum number of data points to 20. The FOR-NEXT loop at lines 140-180 repeatedly invokes the subroutine at line 1000 to enter the data points into the X and Y arrays. The main program is made from lines 200-400. These statements produce a menu which gives the user several paths to follow. The user enters a number from 1 to 5 which is stored in the variable G. The program calculates the appropriate line number from G and branches accordingly. (Note that branching to a line number specified by a variable is one feature of the Sinclair Basic not found in many others. This feature offers many advantages including increased readability; it is much easier to comprehend a statement that says GOSUB PLOTSUB than one that says GOSUB 130.) The options given to the user are: calculating the least squares line, printing a table of the data, plotting the data, starting over or ending the program.

The analytical calculations are done in the subroutine starting at line 3000. As you can see, the summations indicated in the equations in Figures 1 and 2 are carried out in a FOR-NEXT loop from 1 TO N, where N is the number of points. Line 3340 prints the equation of the line from the calculated slope and y-intercept. The correlation coefficient is calculated at lines 3360-3400 and is printed at line 3420. This module, along with some way of getting data into the X and Y arrays, may be used as the main program in a 1K system.

The plotting routine is useful as a qualitative assessment of the data. By plotting the data on the screen one can see at a glance if the data are well behaved: in this case linear. In some cases the user may or may not wish to see a plot which includes the origin. For instance, a straight-line plot with a high positive y-intercept would not utilize the full resolution of the 44 x 64 pixel screen. Instead, the line would look compressed against the top of the screen. For this reason the user is given the option of having the graph include the origin or the point (0,0) which is in the lower left corner of the screen. This is a rather unsophisticated plotting routine and is only valid for positive values. The user may wish to substitute another. A more sophisticated routine would allow negative values and position the origin accordingly. Labeled axes would also be a useful enhancement.

Note the instruction POKE 16437,255 at line 5581. This is necessary for ZX80 8K ROM users. It *must* follow every PAUSE instruction. Failure to include it can result in a system crash. The reason for this is not clear to me, but it is apparently the result of a bug in the ROM. It has something to do with a counter that both the PAUSE instruction and video circuitry use. Without the extra instruction the system may work fine for as long as ten minutes then suddenly produce a blank screen with no response to keyboard input. A power down or reset is the only road to recovery which, of course, results in losing the program which you had just spent the last two hours typing in. I had to find out the hard way, and, after several weeks of testing, probing, and endangering my sanity, I happened to stumble across the solution on page 127 of *ZX81 BASIC Programming*, a typical case of "when all else fails, read the instructions." Be forewarned.

Example

Perhaps the best explanation of the method of "least squares" is an example. Let's say that we are looking for the function which describes the forward voltage drop across a transistor as a function of temperature. In order to use the least squares method we must have a function which we will get back as a straight line. Therefore, we must assume that the relation of the voltage drop across a transistor is linear. Well, it is not quite linear, but close enough when operated between 0 and 100 degrees Celsius. The data in Figure 3 were collected using a 4-digit multimeter and a National Bureau of Standards mercury thermometer with absolute accuracy to 1/100 of a degree.

Figure 3: Voltage/Temperature Table

X Voltage Drop (volts)	Y Temperature (degree Celcius)
(1) 0.6853	0.00
(2) 0.6752	5.00
(3) 0.6647	10.00
(4) 0.6542	15.00
(5) 0.6440	20.00
(6) 0.6337	25.00
(7) 0.6234	30.00
(8) 0.6024	40.00
(9) 0.5811	50.00
(10) 0.5595	60.00
(11) 0.5377	70.00
(12) 0.5264	75.00
(13) 0.5163	80.00
(14) 0.5054	85.00
(15) 0.4942	90.00
(16) 0.4821	95.00

From this table and with the aid of the program in Figure 4 we should easily be able to determine the temperature of the transistor casing (or ambient temperature if the device is in the surrounding environment) by measuring its forward voltage drop. Conversely, we could determine the forward voltage drop if we knew the temperature. Suppose we measure a voltage drop of 0.6295 volts across the transistor and we wish to determine the temperature accurately. We know that it should be between 25 and 30 degrees just by looking at the table. But we want to be a little more accurate than that. Let's run the program and see.

The program first asks for the number of data points. We have collected 16 known data points so enter 16. The program then asks for the data, X1 then Y1, X2 then Y2, etc., until the Y component of the last data point is entered. Then a menu is displayed so we can tell the computer what to do with the data: 1) perform least squares analysis, 2) list the numerical values, 3) plot the data, 4) start over, or 5) quit. The best thing to do here is list the data to double check all of the entries. If a mistake is found, the program forgivingly offers the option of correcting it without entering the entire collection again. After entry corrections, the program waits for another NEWLINE before going back to the menu. With the corrected data now in the computer we are ready to perform a least squares analysis. The program responds with the slope, y-intercept, and the equation of the best straight line through the set of data points. The unknown quantity can be immediately calculated by substituting the corresponding known quantity into the equation.

$$y = ax + b$$

or

$$x = \frac{(y-b)}{a}$$

In this case we know x.

$$y = -469.02146 \text{ deg/V} * 0.6295 \text{ V} + 321.99771 \text{ deg} = 26.75 \text{ deg}$$

The result is valid to the number of digits of the input data. The same results would have been obtained had we switched the coordinates, that is, if we had let the voltage drop be the X value and the temperature be the Y value. Then the unknown temperature would have been

determined by substitution of the transistor's voltage drop into y. Finally, we can plot the data by hitting NEWLINE to get back to the menu and then entering 3. The sample plot reveals that the data are hardly scattered, as also predicted by the correlation coefficient of very near unity.

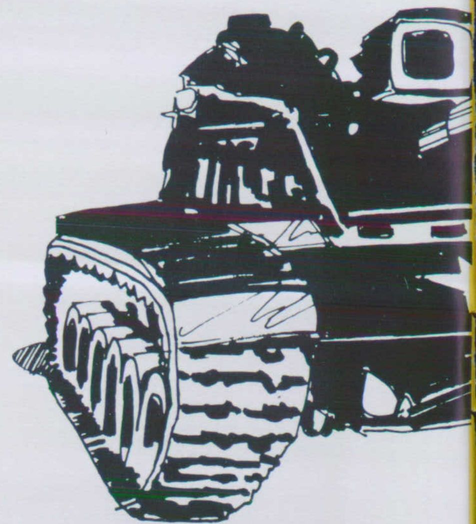
Hopefully, this article will be of help or interest to many readers. This article barely scratches the surface of statistical analysis and function fitting. Bear in mind that the least squares technique is not limited to linear functions. Variations on the method can be used to fit polynomials of any degree to a set of data points.

Note: Readers are referred to Jon Passler's "Linear Regression" in SYNC 2:1 for more information on relating variables.

Figure 4: Program Listing

```
   1 REM ...     LEAST SQURES
   2 REM ... ANALYSIS PROGRAM
   3 REM ...          BY
   4 REM ...      ALGER SALT
   5 REM ...    JANUARY 1982
  10 DIM X(20)
  11 DIM Y(20)
 100 PRINT "LEAST SQUARES DATA A
NALYSIS"
 105 PRINT
 106 PRINT
 110 PRINT "ENTER NUMBER OF POIN
TS"
 120 INPUT N
 121 CLS
 140 FOR I=1 TO N
 160 GOSUB 1000
 180 NEXT I
 200 PRINT
 220 PRINT "1- LEAST SQUARES"
 240 PRINT "2- LIST DATA"
 260 PRINT "3- PLOT DATA"
 270 PRINT "4- START AGAIN"
 280 PRINT "5- QUIT"
 290 PRINT
 300 PRINT "ENTER 1 TO 5"
 320 INPUT G
 321 CLS
 340 GOSUB G*1000+2000
 360 INPUT A$
 361 CLS
 400 GOTO 200
 999 STOP
1000 PRINT AT 10,10;"ENTER X(";I
;")"
1020 INPUT X(I)
1021 CLS
1040 PRINT AT 10,10;"ENTER Y(";I
;")"
1060 INPUT Y(I)
1061 CLS
1099 RETURN
3000 REM ... LEAST SQUARES
3010 LET S0=0
3020 LET S1=0
3040 LET S2=0
3060 LET S3=0
3080 LET S4=0
3100 FOR I=1 TO N
3110 LET S0=S0+Y(I)**2
3120 LET S1=S1+X(I)
3140 LET S2=S2+Y(I)
3160 LET S3=S3+X(I)*Y(I)
3180 LET S4=S4+X(I)**2
3200 NEXT I
3220 LET S5=S1**2
3240 LET M=((S1*S2/N)-S3)/((S5/N
)-S4)
3260 LET B=((-M*S1)+S2)/N
3270 PRINT
3280 PRINT "SLOPE","Y INTCP"
3290 PRINT
3300 PRINT M,B
3310 PRINT
3320 PRINT
3330 PRINT
3335 PRINT "EQUATION OF THE LINE
"
3336 PRINT
3340 PRINT "Y = ";M;"*X + (";B;"
)"
3360 LET SY=SQR ((S0-(S2**2)/N)/
(N-1))
3380 LET SX=SQR ((S4-(S1**2)/N)/
(N-1))
3400 LET CC=M*SX/SY
3410 PRINT
3411 PRINT
3420 PRINT "COR. COEF. = ";CC
3999 RETURN
4000 REM ... LISTING ROUTINE
4005 PRINT AT 0,0;"I   ";"   X","Y
"
4010 PRINT
4100 FOR I=1 TO N
4110 LET R=I+1
4120 PRINT AT R,0;I;
4121 PRINT AT R,3;" ";
4125 PRINT AT R,4;X(I);
4130 PRINT AT R,16;Y(I)
4140 NEXT I
4160 PRINT
4200 PRINT "DO YOU WANT TO CHANG
E ONE?"
4220 INPUT A$
4240 IF A$<>"Y" THEN GOTO 4300
4260 PRINT "WHICH ONE?"
4280 INPUT I
4281 CLS
4290 GOSUB 1000
4295 GOTO 4000
4300 RETURN
5000 REM ... PLOT ROUTINE
5010 LET XL=X(1)
5020 LET YL=Y(1)
5040 LET XH=X(1)
5060 LET YH=Y(1)
5099 REM ...FIND MAX AND MIN X,Y
5400 FOR I=1 TO N
5420 IF X(I)<XL THEN LET XL=X(I)
5440 IF Y(I)<YL THEN LET YL=Y(I)
5460 IF X(I)>XH THEN LET XH=X(I)
5480 IF Y(I)>YH THEN LET YH=Y(I)
5500 NEXT I
5510 PRINT "INCLUDE ORIGIN?"
5511 INPUT A$
5512 CLS
5513 IF A$<>"Y" THEN GOTO 5528
5516 LET XL=0
5517 LET YL=0
5520 FOR I=0 TO 63
5521 PLOT I,0
5523 NEXT I
5524 FOR I=0 TO 41
5525 PLOT 0,I
5526 NEXT I
5528 FOR I=1 TO N
5530 LET XN=(X(I)-XL)/(XH-XL)*41
5540 LET YN=(Y(I)-YL)/(YH-YL)*41
5560 PLOT XN,YN
5580 PAUSE 30
5581 POKE 16437,255
5600 NEXT I
5999 RETURN
6000 GOTO 1
9999 STOP
```

4K ROM
16K RAM
4K RAM



An Alternative Display Method - Tank Battle

Drew Nisbet

Perhaps the greatest inconvenience presented by the ZX80 to the person programming in Basic is the loss of video which occurs whenever the CPU is otherwise occupied. While you are sitting, perhaps not so patiently, waiting for a display to appear, the amount of time that the screen is blank seems much longer than it actually is. This may be only mildly irritating when the program does not depend on user interaction. However, if the program involved is a board game which requires that an ever-changing playing arena be displayed repeatedly and that the program interact with the player to some degree, then periods of time spent before an unimpressive, grey screen can be fatal to the attention span of the player.

The majority of board games written in Basic for the ZX80 display the playing surface by storing print codes in an integer array and printing each member of this array by means of a FOR/NEXT loop. Any change on the board, such as the movement of a token or the destruction of a coordinate, requires that the screen be cleared and that the display be re-created. Generally speaking, the larger the playing surface of a game, the better, but displaying a board which uses a large portion of the screen area by the method noted above can take up to 30 seconds. Add to this the time required by the computer to perform any calculations which are required between displays and you may wind up with the plug being pulled on your program! In addition, any prompts or messages displayed for the player's information must be kept to a minimum, or the playing surface must again be cleared in order to accommodate them.

Drew Nisbet, 6 Moffatt Court, Toronto, Ontario, Canada, M9V 4E1

Figure 1: Token Use Directions.

Symbol	Meaning
+	Available coordinate
▒	Destroyed coordinate
H	Location of player's tank
C	Location of computer's tank °
Inverse *	Player's tank destroyed °
Inverse C	Computer's tank destroyed °

° Displayed at end of game

Although the most valuable use of the PEEK and POKE commands may be to program and debug ZX80 machine code programs, they are extremely valuable when employed in any Basic program as tools for both examining and altering memory location contents. After an initial display has been created on the screen, it may be examined or altered by PEEKing or POKEing memory locations relative to the address stored in the D-FILE pointer (located in the two bytes starting at address 16396). The game program which follows makes extensive use of this function.

The screen display for *Tank Battle* is divided into two sections: the playing surface, or board, which occupies two thirds of the display, and a "message center" which makes use of the remaining third of the screen and is located to the right of the game board. This arrangement affords the largest square playing area available. The two sections are prominently separated by a black line. The PRINT command is not employed after the initial playing board has been displayed as all alterations to the playing surface are made by POKEing character codes into memory. All prompts and messages are displayed and erased in the same manner. The longest period of time during which the screen is blank is approximately five seconds, the amount of time depending on the number of lines which must be deleted from the message portion of the screen.

Figure 3: The Tank Battle Program.

```
 10 RANDOMISE
 20 LET HO = RND(9) * 10 + RND(9)      player's last coord=dummy
 30 LET HV = RND(9) * 10 + RND(9)      player's 2nd last coord=dummy
 40 LET HN = 0                         player's last move
 50 LET CM = 1                         computer's move
 70 LET SH = 0                         computer's target
 80 LET A$ = "COORD:"
 90 LET B$ = "YOU WIN"
100 LET C$ = "I WIN"
110 LET D$ = "I FIRED"
120 LET E$ = "FROM"
```

"#" represents a space

"@" stands for the character shifted "Q".

Figure 2: Screen Display after Several Moves.

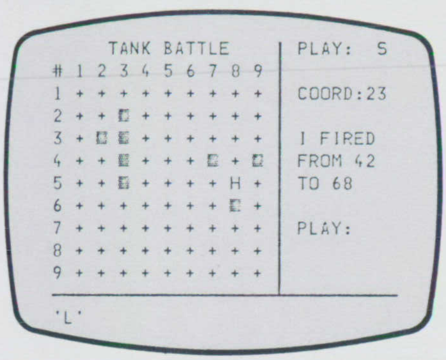


Figure 2 shows the screen layout after both the player and the computer have taken several turns. The drawing is not to scale as the playing board is really double spaced both vertically snd horizontally. The . which does not print on the screen. is the point from which all PEEKing and POKEing is done in order to examine or alter the board contents. The address stored in the D-FILE pointer and 67 (32 print codes plus 1 newline character times 2 plus 1 print code on the third print line equals 67). The address for any coordinate on the board is determined by multiplying the row coordinate by 6 (2 times 32 print codes plus 1 newline character). adding 2 times the column coordinate (for the horizontal double spacing) and adding 67 (for the reference point) to the contents of the D-FILE pointer. Alterations to the message portion are accomplished by POKEing the appropriate character codes into memory locations relative to the contents of the D-FILE pointer.

The program uses the subroutines at lines 1600 and 1910 to accomplish the above. The subroutine at line 1600 alters any memory location by POKEing a character code M into the location relative to the contents of the D-FILE pointer by an offset of P locations. The subroutine at line 1910 POKEs the message codes into consecutive memory locations by

Figure 3 (continued)

```
130 LET F$ = "TO"
140 LET G$ = "PLAY:"
150 LET H$ = "TIE GAME"
160 LET I$ = "GAME OVER"
170 LET J$ = "AGAIN?"
180 LET D = 5                            player's beginning row
190 LET E = 8                            player's beginning column
200 LET HL = 58                          player's beginning coord
210 LET CL = RND(9) * 10 + 1
220 PRINT "#####TANK BATTLE####@#PLAY:#####"
230 PRINT "####################@###########"
240 PRINT "#";
250 FOR I = 1 TO 9
260 PRINT "#";I;
270 NEXT I
280 PRINT "#@###########"
290 PRINT "####################@###########"
300 FOR I = 1 TO 9
310 PRINT I;"#+#+#+#+#+#+#+#+#+##@###########"
320 PRINT "####################@###########"
330 NEXT I
335 FOR I = 1 TO 32
336 PRINT CHR$(131);
337 NEXT I
340 LET P = 67 + 66 * D + E * 2
350 LET M = 45                           'H'
360 GO SUB 1600                          print player's location
370 INPUT M$                             input play
380 IF NOT (M$ = "M" OR M$ = "S" OR      if invalid ask again
    M$ = "") THEN GO TO 370
390 IF M$ = "" THEN GO TO 800            if end of game branch
400 LET P = 29                           print play selected
410 LET M = CODE(M$)
420 GO SUB 1600
430 LET P = 89                           print 'COORD:'
440 LET T$ = A$
450 GO SUB 1910
460 INPUT C                              input coord
470 LET D = C / 10                       row
480 LET E = C - D * 10                   column
490 IF D < 1 OR D > 9 OR E < 1 OR E > 9  if invalid coord ask again
    THEN GO TO 460
500 IF M$ = "S" THEN GO TO 660           if shoot branch
510 LET G = HL / 10                      player's row
520 LET H = HL - G * 10                  player's column
530 IF ABS(G - D) > 1 OR ABS(H - E) > 1  if illegal distance ask again
    THEN GO TO 460
540 LET P = 67 + 66 * D + E * 2
550 IF PEEK(PEEK(16396) + PEEK(16397)    if illegal coord ask again
    * 256 + P) = 128 THEN GO TO 460
560 GO SUB 1620
570 LET P = 67 + 66 * G + H * 2
580 LET M = 19                           '+'
590 GO SUB 1600                          blank player's old position
600 LET HL = C                           coord = new coord
610 LET P = 67 + 66 * D + E * 2
620 LET M = 45                           'H'
630 GO SUB 1600                          display player's new position
640 LET HM = 1                           play = move
650 GO TO 890                            branch to computer's turn
660 GO SUB 1620                          display coords
670 LET P = 67 = 66 * D + E * 2
680 LET M = 128                          '▪'
690 GO SUB 1600                          display hit coord
700 LET HM = 2                           play = shoot
710 LET HO = HL                          let old coord = present coord
720 GO TO 890                            branch to computer's turn
730 LET P = 353
740 LET T$ = J$                          print 'GAME OVER'
750 GO SUB 1910
760 INPUT Y$                             input yes or no
```

examining the strings (T$) one character at a time. Lines 1430 - 1560 POKE zeros (code for a space) into any line which contains non-blank characters in the message portion of the screen. The routines located at lines 1620, 1690, 1970, and 2010 display the coordinates selected by the player, the shots fired by the computer, the player's tank when destroyed and the computer's tank when destroyed respectively.

The game itself was inspired by Lloyd Johnson's article "Paint Duel" in *Creative Computing* (July 1981). The player attempts to destroy the computer's tank by hitting it with a well-placed shot. The two plays available are Shoot or Move. The player's initial location is row 5 column 8. The computer tries to destroy the player's tank in turn. The starting location for the computer's tank is in the first column and a random row. After specifying the play selected the computer requests the coordinates of the target and then displays them in the message portion of the screen. A shot shows up on the screen as a "□" placed at the coordinates specified. A move is legal only if made to a coordinate directly above, below, left of, right of, or diagonally adjacent to the present location. A tank may not move to a coordinate which has previously been shot at.

The computer's play is determined by the following odds:

1) 10% chance it will do nothing.
2) 10% chance it will shoot.
3) 70% chance it will move if it shot last turn.
4) 70% chance it will fire if player shot last turn.
5) Move.

These odds can be altered by changing the values in lines 890 - 920 of the program. The computer is prevented from firing at a coordinate which has already been shot at by PEEKing into the memory location for the coordinate selected and comparing its contents to 128 (black square). The tanks are prevented from moving into a hit square in the same way. Both the player's and the computer's plays are completed simultaneously, which allows the possibility of a tie game if both tanks are destroyed on the same turn. Each player learns of the position of the opponent's tank only if he is fired upon. If the computer decides to shoot, it will place its shot into a coordinate which it knows the player may have moved to. To end the game press NEWLINE when requested to play.

Ed.—For those who do not wish to enter the listing, it is available from the author on cassette. Write for information.

Figure 3 (continued)

```
 770 IF Y$ = "N" THEN STOP                    stop if no replay
 780 CLS
 790 RUN                                      run if replay
 800 LET D = CL / 10                          display computer's location
 810 LET E = CL - D * 10
 820 LET P = 67 + 66 * D + E * 2
 830 LET M = 40
 840 GO SUB 1600
 850 LET P = 287
 860 LET T$ = I$                              print 'GAME OVER'
 870 GO SUB 1910
 880 GO TO 730                                ask for replay
 890 LET X = RND(10)
 891 IF X > 9 THEN GO TO 1140                 10% chance to do nothing
 892 IF X < 2 THEN GO TO 1020                 10% chance to shoot
 900 IF CM = 2 AND RND(10) > 3 THEN           70% chance to move
     GO TO 920
 910 IF HM = 2 AND RND(10) > 3 THEN           70% chance to shoot
     GO TO 1020
 920 LET D = CL / 10                          computer moves
 930 LET E = CL - D * 10
 940 LET S = D + RND(3) - 2
 950 LET T = E + RND(3) - 2
 960 IF S < 1 OR S > 9 OR T < 1 OR            if invalid coord
     T > 9 THEN GO TO 940
 970 LET P = 67 + 66 * S + T * 2
 980 IF PEEK(PEEK(16396) + PEEK(16397)        if illegal coord
     * 256 + P) = 128 THEN GO TO 900
 990 LET CL = S * 10 + T                      computer's new location
1000 LET CM = 1                               computer's play = move
1010 GO TO 1140                               branch to request next play
1020 LET D = HV / 10                          computer shoots
1030 LET E = HV - D * 10
1040 LET S = D + RND(3) - 2
1050 LET T = E + RND(3) - 2
1060 IF S < 1 OR S > 9 OR T < 1               if invalid coord
     OR T > 9 THEN GO TO 1040
1070 LET P = 67 + 66 * S + T * 2
1080 IF PEEK(PEEK(16396) + PEEK(16397)        if coord already hit
     * 256 + P) = 128 THEN GO TO 1040
1090 LET SH = S * 10 + T                      coord of target
1100 LET M = 128                              '□'
1110 GO SUB 1600
1120 GO SUB 1690                              display computers shot
1130 LET CM = 2                               computers play = shoot
1140 IF HM = 2 AND C = CL THEN GO TO 1170       if computer's tank hit
1150 IF CM = 2 AND SH = HL THEN GO TO 1230      if player's tank hit
1160 GO TO 1390                                 if no hits
1170 IF CM = 2 AND SH = HL THEN GO TO 1330      if both hit
1180 GO SUB 1970                              display computer's hit tank
1190 LET P = 287
1200 LET T$ = B$                              print 'YOU WIN'
1210 GO SUB 1910
1220 GO TO 730                                branch to ask for replay
1230 GO SUB 2010                              display player's hit tank
1240 LET P = 287
1250 LET T$ = C$                              print 'I WIN'
1260 GO SUB 1910
1270 LET D = CL   10                          display computer's location
1280 LET E = CL - D * 10
1290 LET P = 67 + 66 * D + E * 2
1300 LET M = 40                               'C'
1310 GO SUB 1600
1320 GO TO 730                                branch to ask for replay
1330 GO SUB 1970                              display computer's hit tank
1340 GO SUB 2010                              display player's hit tank
1350 LET P = 287
1360 LET T$ = H$                              print 'TIE GAME'
1370 GO SUB 1910
1380 GO TO 730                                branch to ask for replay
1390 LET P = 287
```

Figure 3 (continued)

```
1400 LET T$ = G$                          print 'PLAY:'
1410 GO SUB 1910
1420 INPUT M$                             input play
1430 LET M = 0
1440 LET P = 29                           blank out old play message
1450 GO SUB 1600
1460 LET P = 89                           blank out all messages
1470 IF NOT PEEK(PEEK(16396) + PEEK(16397)     lines which are non-blank
     * 256 + P) = 0 THEN GO TO 1500
1480 LET P = P + 33
1490 GO TO 1550
1500 FOR I = 1 TO 8
1510 POKE PEEK(16396) + PEEK(16397) * 256 + P,0
1520 LET P = P + 1
1530 NEXT I
1540 LET P = P + 25
1550 IF P > 320 THEN GO TO 1570
1560 GO TO 1470
1570 LET HV = HO                          2nd last location = last
1580 LET HN = HM                          last play = play
1590 GO TO 380                            go display next play
1600 POKE PEEK(16396) + PEEK(16397)       poke character code in memory
     * 256 + P,M
1610 RETURN
1620 LET P = 95                           poke coords in message section
1630 LET M = D + 28
1640 GO SUB 1600
1650 LET P = P + 1
1660 LET M = E + 28
1670 GO SUB 1600
1680 RETURN
1690 LET P = 155                          display computer's shot
1700 LET T$ = D$                          print 'I FIRED'
1710 GO SUB 1910
1720 LET P = 188
1730 LET T$ = E$                          print 'FROM'
1740 GO SUB 1910
1750 LET P = 193
1760 LET M = CL / 10 + 28                 print coords
1770 GO SUB 1600
1780 LET M = CL - (CL / 10 ) * 10 + 28
1790 LET P = P + 1
1800 GO SUB 1600
1810 LET P = 221
1820 LET T$ = F$                          print 'TO'
1830 GO SUB 1910
1840 LET P = 224
1850 LET M = S + 28                       print coords
1860 GO SUB 1600
1870 LET M = T + 28
1880 LET P = P + 1
1890 GO SUB 1600
1900 RETURN
1910 LET M = CODE(T$)                     print first character of
1920 GO SUB 1600                                    string
1930 LET T$ = TL$(T$)                     shorten string
1940 IF T$ = "" THEN RETURN               if all done return
1950 LET P = P + 1
1960 GO TO 1910                           branch to print next character
1970 LET P = 67 + 66 * (CL / 10) + 2      print computer's hit tank
     * (CL - (CL / 10) * 10)
1980 LET M = 168                          inverse 'C'
1990 GO SUB 1600
2000 RETURN
2010 LET P = 67 + 66 * (HL / 10) + 2      print player's hit tank
     * (HL - (HL / 10) * 10)
2020 LET M = 148                          inverse '*'
2030 GO SUB 1600
2040 RETURN
SYNCSUM = 123       4K ROM        4K RAM
```

resources

Users Groups

Central Pennsylvania Interest Group
Meets every Tuesday at 8:00 p.m. 321 Bouche Bldg., Penn State U. Campus, State College, PA. Initiator: Bill Russell. Phone: 814-364-1325.

Club ZX 80/81
Chemin du Moulin 38
B - 1328 OHAIN
Belgium

SAM BAM (Sinclair and MicroAce Bay Area Microcomputer Users). Users "living in the Golden Triangle around Tampa Bay and along the golden suncoast of Florida" are invited to write for information. Locating users in the area is the immediate task.
SAM BAM
c/o Mel Routt
PO Box 596
Safety Harbor, FL 33572

Memphis, TN, ZX81 Users Group just getting started. Contact:
James Barker
3791 Barron Ave.
Memphis, TN 38111
Phone: (901) 327-2158

Voltage devices

VOTEM for ZX81/ZX80, 8K ROM.
Read voltage and temperature with extremely high sensitivity. Voltage range easily adjusted for wide variety of applications; temperature range: -25 to 125 C; self-contained and low power unit attaches directly to ZX80/81 with no hardware modifications; all calibration done in software; audio and visual tape load monitoring capability; circuitry for amplifying and conditioning tape signal. Software runs in 1K RAM. Kit, $39.95; assembled and tested, $59.95.
Down East Computers
PO Box 3096
Greenville, NC 27834

Protect all ZX80/81 ICs if voltage regulator fails by crowbar and fusing. Schematic, $2.00; with parts, $15.00. Schematic for precision setting of cassette output level, $2.00.
Arnold J. Gold
820 Prospect Ave.
Hartford, CT 06105

Books

Word Processors & Information Processing: A Basic Manual on What They Are and How To Buy, by Dan Poynter. $11.95 from:
Para Publishing
PO Box 4232-88
Santa Barbara, CA 93103-0232
Phone: (805) 968-7277

Memory Expansion

RKL-16K RAM, $79.95; expandable to 32K whenever desired. RKL-32K RAM, $149.95. Upgrade 16K to 32K, $79.95. Add $4.95 s&h and 5% tax in Mass.
RKL Systems
PO Box 515
Leominster, MA 01453

Joysticks

RKL-J1, single joystick system, $69.95; RKL-J2, dual joysticks, $89.95. Upgrades: RKL-U2, adds RKL-J1 to RKL-32, $59.95; RKL-U3 adds RKL-J2 to RKL-32, $79.95. Add $4.95 s&h and 5% tax in Mass.
RKL Systems
PO Box 515
Leominster, MA 01453

Publications

SYNCHRO-SETTE, a monthly magazine with bi-monthly cassette containing at least six programs for the ZX81 or ZX80(8K ROM) user. One year, $39.50. Visa and MC. For more information call 24 hr. toll-free hotline, 800/543-1300 (in Ohio, 800/582-1364), ask for operator 383.
S & S Company
388 W. Lake St.
Addison, IL 60101
Phone: 312/628-8955

Micro Moonlighter Newsletter will provide techniques, tips, and guidance to those who wish to make money with their computer knowledge. Among the topics: Marketing methods, advertising hints, the ins and outs of contracts, books. Subscriptions: $25 for 12 issues ($29 in Canada; $35 worldwide). Visa and Master Charge welcome. Send inquiries to:
J. Norman Goode
2115 Bernard Ave.
Nashville, TN 37212

ZX81 Sourcebook (TM)
A catalogue listing software and hardware for the ZX81. For information on getting listed and getting copies, contact:
Micro Design Concepts
PO Box 280
Carrollton, TX 75006

The Index covers over 40 magazines and newsletters in the home/personal computer marketplace and includes over 12,000 articles indexed by using the key words of the titles. For information and ordering contact:
Missouri Indexing, Inc.
P.O. Box 301
St. Ann, MO 63074
Phone: (314) 997-6470

ROM Disassembly

The ZX80 1K Disassembler for the 4K ROM ZX80.
The ZX81 1K Disassembler for the ZX81 and 8K ROM ZX80.
To begin disassembly of ROM or MC program, enter starting address, a key is calculated and displayed, manual then give mnemonic, all numbers displayed in decimal, about 100 bytes of RAM to store MC program for disassembly; RAM memory test provided; addresses of bytes failing the test are displayed. Manual, reference cards, and cassette with the disassembler and memory test programs. $9.95 pp. Further information and catalog upon request.
Lamo-Lem Laboratories
Box 2382
La Jolla, CA 92038-2382

Programs

8K ROM; 16K, 1K. Games, utilities, tech prog., household, and more. For details send SASE to:
NGM INC.
PO Box 18702
Okla City, OK 73154

ZX Galaxians

Martin Wren-Hilton

SYNC

SOFTWARE PROFILE

Name: ZX-Galaxian

Type: Arcade fantasy

System: ZX81; ZX80 8K ROM, SLOW mode; 4K RAM

Format: Cassette

Language: Z80 Machine Code

Summary: A challenging game and a good implementation of the arcade game

Price: £3.95; £1.00 s&h for U.S.

Manufacturer:
Artic Computing
396 James Reckitt Avenue
Hull, N. Humberside HU8 OJA
United Kingdom

ZX-Galaxians is a good adaptation to the ZX81 of the popular arcade game of the same name. You are being attacked in deep space by formations of hostile Galaxians, and it is your mission to prevent them from attacking Earth by zapping them with your laser gun as they break out of formation and hurtle towards you.

After being loaded, the game runs automatically. The title appears in large letters at the top of the screen, a bit of information about the game is shown and you press any key to start. Four rows of eight galaxians appear hovering above your base. On the right side of the screen you will see the details of which buttons to press, what the current score is, what the high score is together with the name (up to six letters) of the high scorer, and a graphical representation of how many ships you have left.

The thirty-two galaxians move left to right and back again. Your controls are 5 for left, 8 for right and 0 for fire. Occasionally one or more galaxians break out of the formation and dive about the screen, dropping bombs as they go. You get 10 points for each galaxian shot in formation and 20 points for those shot in mid-flight. The movement of your base is very smooth due to good use of the ZX81's graphics. When your base gets hit either by a galaxian or by a galaxian's bomb, the explosions are quite good.

Unlike the real Galaxians (TM), this game does not have the starry background. In addition the formation is rectangular. The letter "V" is used to represent the galaxians, and those flying around the screen are made up from the graphic symbols on letters Q, W, E and R, depending upon their direction of flight.

If you are a keen arcade player, then *ZX-Galaxians* is for you. It is the best version of this game that I have played, and makes good use of the ZX81's graphics.

Martin Wren-Hilton, U.K. Correspondent to SYNC, 4 Little Poulton Lane, Poultonle-Fylde, Backpool, FY6 7ET, United Kingdom.

4K ROM
1K RAM

SPACE WARP—A Graphics Space Game

Armando Fox

Since I have always loved video space games, I have been looking for some game that could possibly fit into my limited 1K of RAM. The product of several hours of often exasperating labor is *Space Warp*, which runs in 1K without any problems if entered as specified.

The objective of *Space Warp* is to reach your base before running out of fuel. In order to do this, you must think ahead and consider how your speed will affect your fuel consumption. The distance to the base is 1000 miles, and you have 100 fuel units initially. Your speed, measured in warp factors as per Star Trek, has a direct bearing on your fuel consumption. Warp factors range from 1 (slowest) to 15 (fastest). Each turn, you will be asked to enter a warp factor. A read-out of fuel remaining, distance remaining, velocity, and warp factor as of last turn is also displayed. After entering the warp factor, you will be asked "DIRECTION?" Input F to move forward towards the base or R to reverse. The reason for reversing will be discussed below. When this has been done, one of two things will happen.

If you are unlucky, an enemy satellite dispatched from a nearby base will harass your approach. This will appear as a double asterisk (**) at the rightmost edge of the screen. If you do not watch out for these satellites, they "eat" a random number of fuel units. This is often the downfall of many players. However, if you are going at warp 7 or less, you will be asked "RETREAT?" Answer Y or N. If you retreat, you will of course lose distance as well as fuel, but the enemy satellites cannot harm a retreating ship, so retreating does pay off as a strategy technique. If you decide to retreat, the appropriate fuel (and distance) will be added into the next calculation; if you decide not to, the enemy satellite will "eat" a random number of fuel units.

Next (or first, if no satellite came after you) the screen will clear and then redisplay, hopefully with your ship farther over to the right, indicating that you have moved closer to the base. The readout will again be displayed, and the "WARP" and "DIRECTION" prompts will wait for input.

This turn sequence continues until one of three things happens:

1) You reach the base and still have fuel, and the message "HOME FREE" will appear near the center of the screen.

2) You will reach the base without any fuel, drift, and collide with it—your ship will break up into several pieces.

3) You run out of fuel before reaching the base, and the game will stop with error code 9/460. If you want to know how far you actually travelled, subtract the distance on the screen from 1000.

Entering the Program

This program runs in 1K, but it must be entered in two parts. The first part is the "set variables" section shown in Listing 1. Enter this short program and RUN. Enter the following values to be POKEd to the REM statement in line 1: 0, 3, 10, 0, 0, 132, 128, 150, 139, 10, 0, 132, 3, 133, 6. Then hit LIST. Line 1 will look like a jumble of graphics symbols; these draw the ship. Now delete lines 40 through 70 (*not* 1 through 30) and continue by entering the main program in Listing 2. To run the program, simply type RUN; since the graphics are stored as a REM statement, there is not much that can cause these variables to be cleared, except for NEW. ■

Armando Fox, 62-23 Cromwell Crescent, Rego Park, NY 11374.

Listing 1: Setting the Variables.

```
 1 REM XXXXXXXXXXXXXXX  (15 X's)
10 LET D=1000
20 LET D2=0
30 LET F=100
40 FOR X=16427 TO 16441
50 INPUT P
60 POKE X, P
70 NEXT X
```

Listing 2: Space Warp.

```
100 PRINT "WARP?"
110 INPUT W
120 IF W>15 OR W<1 THEN GO TO
    110
130 PRINT "DIRECTION?"
140 INPUT D$
150 IF NOT (D$="F" OR D$="R") T
    HEN GO TO 140
200 LET F=F-W*RND(3)
210 LET V=(W/2)*(RND(5)+5)
220 LET D1=V*(W/2)
230 IF D$="R" THEN LET D1=-D1
240 LET D=D-D1
250 LET S=0
260 IF RND(6)=6 THEN LET S=226
270 LET D2=D2+D1
280 IF D<0 THEN LET D=0
290 CLS
300 PRINT "WARP", "FUEL", "VELO
    C", "DIST", W, F, V, D
320 PRINT
330 FOR X= -4 TO 6
340 FOR J=1 TO D2/40
350 PRINT "#";
360 NEXT J
370 FOR N=X+5 TO X+9
380 PRINT CHR$ (PEEK (16426+N));
390 NEXT X
400 PRINT
410 LET X=X+4
420 NEXT X
425 FOR L=1 TO 25
430 PRINT "#";
435 NEXT L
440 PRINT CHR$ (S)
450 IF S THEN GO SUB 600
460 IF F<1 THEN STOP
470 IF D<1 THEN GO TO 500
480 GO TO 100
500 PRINT , "HOME FREE"
510 STOP
600 IF W>7 THEN GO TO 660
610 PRINT "RETREAT?"
620 INPUT D$
630 IF D$="N" THEN GO TO 660
640 LET D=D+V*(W/2)
645 LET F=F-W
650 GO TO 670
660 LET F=F-(RND(5)*3+W)
670 RETURN
```

Sample Run

```
WARP?
6 (NL)
DIRECTION?
F (NL)

(screen cleared)
WARP     FUEL     VELOC     DIST
6        82       27        919
```

```
WARP?
10 (NL)
DIRECTION?
F (NL)

(screen cleared)

WARP     FUEL     VELOC     DIST
10       52       40        719
```

4K ROM
1K RAM

Storing 3-Letter Words in an Array

F. W. Manders

I wanted to develop a method of storing as many three-letter words as possible in my 1K ZX80, using as little memory as possible. The solution is based on the fact that character numbers of the letters can be compounded into a single number, which can be stored in an array element. Thus three letters can be stored in two bytes.

Listing 1, *The Word Storing Routine*, sets up arrays which are used in Listing 2, *The WORDY Program*.

The technique takes the character numbers of the letters in a word; subtracts 36, then multiplies by 1 for the first letter, 30 for the second letter, and 900 for the third. Then the three are added together so that they can be stored as a single number, which must not exceed 32767, hence the -36 in line 14. Line 18 ensures that a space keeps its code number of 0, and a nul string is treated as a space. This allows one- or two-letter words to be stored if desired.

Line 20 does the multiplication and addition, and the numbers representing the words are stored in Array S(). The maximum number of words this program can hold is 175.

When the words have been stored, the whole of the word storing routine can be deleted. But remember never use RUN, or the words will be lost. The words could be stored at this stage, but it is better to get as much as possible of the main program entered first.

Having entered Listing 1 and checked the SYNCSUM, which should be 18, enter Listing 2, starting at Line 32. The first part of Listing 2 cannot be entered until you are finished with Listing 1. Check the SYNCSUM again at the end of Listing 2. It should be 57.

Now you are almost ready to store your 175 words, but first the SYNCSUM Routine must be removed as we need the 27 bytes it occupies. We also need to store the program on tape at this stage, so we can kill two birds with one stone, as they say. When you have it safely on tape, enter NEW and then reLOAD from the tape.

Enter RUN and the screen will display:

```
ENTER WORDS
0
```

Now enter a 3-letter word (e.g., AND) and the screen will display:

```
0          AND
1
```

Continue entering words until all 175 have been entered. Then the error code 9/26 will be displayed. Press any key and the program listing will appear. Delete lines 1 to 26 and enter lines 10 to 30 of Listing 2.

When you have saved the completed program on tape, you are ready to start playing "WORDY," which is a 3-letter word

Listing 1: The Word Storing Routine

```
1 DIM A(2)
2 DIM B(2)
3 DIM C(2)%
4 DIM S(174)
5 PRINT "ENTER WORDS"
6 FOR I=0 TO 174
7 PRINT I
8 INPUT U$
9 CLS
10 PRINT I, U$
12 FOR J=0 TO 2
14 LET A(J)=CODE(U$)-36
16 LET U$=TL$(U$)
18 IF A(J)< 2 THEN LET A(J)=0
20 LET S(I)=S(I)+A(J)*30**J
22 NEXT J
24 NEXT I
26 STOP
        SYNCSUM = 18
```

Listing 2: The WORDY Program

```
10 LET W=RND(175)-1
12 LET W=S(W)
14 FOR I=0 TO 2
16 LET J=2-I
18 LET K=30**J
20 LET A(J)=W/K
22 LET W=W-A(J)*K
24 IF A(J)>1 THEN LET A(J)=
   A(J)+36
26 NEXT I
28 PRINT "WORDY"
30 PRINT
32 FOR J=1 TO 10
34 LET D=0
36 INPUT U$
38 PRINT U$;"#";
40 FOR I=0 TO 2
42 LET C(I)=CODE(U$)
44 LET U$=TL$(U$)
46 LET B(I)=A(I)
48 NEXT I
50 FOR I=0 TO 2
52 IF C(I)-B(I) THEN GOTO 62
54 LET D=D+1
56 LET C(I)=3
58 LET B(I)=2
60 PRINT "*";
62 NEXT I
70 FOR I=0 TO 2
72 FOR K=0 TO 2
74 IF C(K)-B(I) THEN GOTO 82
76 LET C(K)=3
78 LET B(I)=2
80 PRINT O;
82 NEXT K
84 NEXT I
86 PRINT
87 IF D=3 THEN GOTO 90
88 NEXT J
90 PRINT
92 FOR I=0 TO 2
94 PRINT CHR$(A(I));
96 NEXT I
98 PRINT "#"; J
```

F. W. Manders, 24 Horton St., Lincoln, LN2 5NG, UK.

version of "Mastermind." Enter GOTO any number up to 10, but not RUN or the words will be lost. Enter your word and NEWLINE. If you think the answer might be a 2-letter word, enter a space for the third letter.

The program uses the multiplication sign to indicate a direct match and the zero for an indirect match. It allows up to 10 tries and prints out the correct answer and the number of tries needed, showing 11 if you have not solved it by the tenth try.

The word selection and unscrambling routines are contained in lines 10 to 26. The selected number is divided by 900, i.e., 30**2, to find the character number of the third letter. Then the remainder is divided by 30 for the second. The final remainder gives the first—when they have each had 36 added at line 24. The rest of the program is, I think, fairly conventional, but note the Boolean Logic in lines 52 and 74.

There are a lot more than 175 3-letter words in the English language, so you might like to make several versions.

Besides showing a method of storing 3 characters in 2 bytes, this program also shows that once arrays have been set up, the routine which established them can be deleted so long as CLEAR and RUN are avoided.

4K ROM
1K RAM

Character Reverse

Daniel Freeman

Have you ever wanted to put a reverse character in a string but you had no way of doing it? Here is a simple answer, a small machine language program that will reverse a character for you:

Decimal	Mnemonic
221,42,8,64	LD IX (nn)
33,70,64	LD HL, nn
35	INC HL
126	LD A (HL)
221,190,0	CP (IX+d)
200	RET cc
254,27	CP n
194,50,64	JP cc, nn
35	INC HL
126	LD A (HL)
198,128	ADD A, n
119	LD (HL) A
195,50,64	JP nn

How It Works

The first line loads the IX register with the memory location of the end of the Basic program, pointed by VARS, then the HL register is loaded, LD HL, nn, with the end of the machine language program. The HL register is then incremented in the line INC HL. In the next line the A register is loaded with the memory contents pointed to by HL. The A register is then compared to the memory location pointed by the IX register. If they are equal, the program returns to Basic. The two will only be equal if it is at the end of the Basic program. The A register is then compared, CP n, with 27, the character number of ".". If they are not equal, then the program jumps up to INC HL and repeats the cycle. If they are equal, then HL is incremented and the A register is loaded with the memory contents pointed by HL. Then 128 is added, ADD A, n, to A to get the reverse of the character. It is then loaded back into the memory location pointed by HL. Finally the program jumps back to the first INC HL and continues until the end of the Basic program.

Daniel Freeman, 2902 E. Hoover Ave., Orange, CA 92667.

Writing the Program

Type the following:

```
 REM 12345678901234567890
 123456
100 FOR X=16427 TO 16452
110 PRINT X,,
120 INPUT A
130 IF PEEK(16421)<2 THEN CLS
140 POKE X,A
150 PRINT PEEK(X)
160 NEXT X
```

Then type in:

10 REM THEN THEN THEN THEN ...

Continue typing THEN until line 1 disappears, then enter:

10 REM DO NOT GO UP

RUN and enter the following values:

16427	221
16428	42
16429	8
16430	64
16431	33
16432	70
16433	64
16434	35
16435	126
16436	221
16437	.190
16438	0
16439	200
16440	254
16441	27
16442	194
16443	50
16444	64
16445	35
16446	126
16447	198
16448	128
16449	119
16450	195
16451	50
16452	64

DO NOT PRESS LIST OR HOME. THE PROGRAM WILL BE LOST IF SHOWN.

Delete lines 100-160; then save the program.

To Operate:

Before writing your own program, load this one. Whenever you need the reverse of a character, insert a "." before the character to be reversed. Then enter GOTO USR(16427). More than one character can be done at a time.

If you wish to use a different character than ".", POKE 16641, (the character number of your choice).

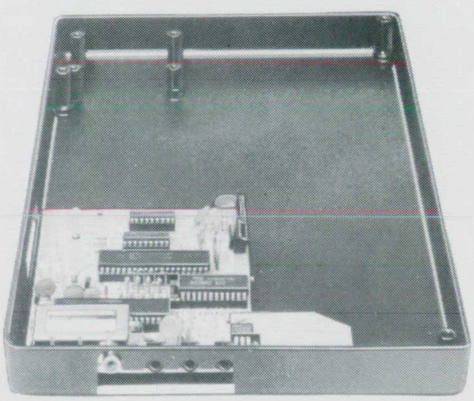
index to advertisers

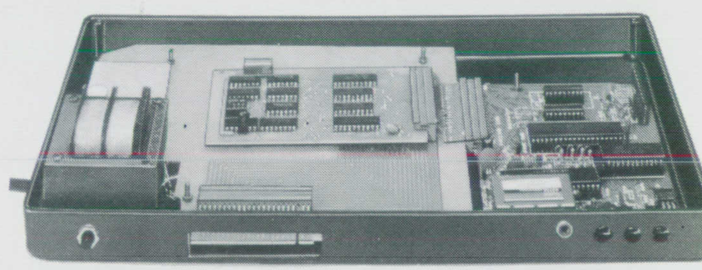
FULLER FD SYSTEM $79.95

Professional Keyboard & Case for Sinclair ZX81

The ZX8l fits inside.
The tough ABS injection moulded plastic case measures 8″ x 14″ x 2½″ and hooks up to your ZX printed circuit board in minutes. No technical know how or soldering is required.

KEYBOARD LAYOUT:
All the Sinclair ZX81 keys are duplicated on our layout, with extra shift and new line keys. The professional momentary action key switches have a guaranteed life of 10^6 operations. The unit is fully built tested and comes complete with a money back guarantee.





INSTALLATION
Simply unscrew the ZX printed circuit board from its case and screw it into the FD Case.

MOTHERBOARD:
We also manufacture a mother board which allows expansion to the ZX memory and I/0 facilities WITHIN the case, as well as our power supply unit and reset switch.

AD Code

		Price †	Qty	Amount
Fuller	FD System 42 Keyboard & case	79.95		
	FD System Motherboard	29.95		
	FD 16K. Memory Module	49.95		
	FD 64K. Memory Module	149.96		
	FD PSU 9 Volts at 2 amp.	14.95		
	FD *Shipping and Handling*	6.00		

Mail to **FULLER MICRO SYSTEMS,**
The ZX Centre, Sweeting Street, Liverpool 2. England, U.K.

Name ....

Address ....

City/State/Zip ....

† US Dollars